Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное
учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Институт естественных и точных наук
Факультет математики, механики и компьютерных технологий
Кафедра прикладной математики и программирования
Направление подготовки: 01.03.02 Прикладная математика и информатика

РАБОТА ПРОВЕРЕНА
Рецензент, зав.каф. «Информа-
ционные технологии в экономике»
с.н.с., д.т.н.
_____/Б.М. Суховилов
«____»_____20___г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой, д.ф.-м.н.,
профессор
_____/А.А.Замышляева
«____»_____20___ г.

Разработка программы «Аренда апартаментов»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ–01.03.02.2020.076.ПЗ ВКР

Руководитель работы, доцент
Кафедры ПМиП, к.т.н.
_____/М.Ю. Катаргин
«____»_____2020 г.

Автор работы
Студент группы ЕТ-413
_____/И.Э. Абдулина
«____»_____2020 г.
Нормоконтролер,
ст. преподаватель
_____/Н.С. Мидоночева
«____»_____2020 г.

Челябинск
2020

АННОТАЦИЯ

Абдулина И.Э. Разработка программы «Аренда апартаментов». – Челябинск: ЮУрГУ, ЕТ-413, 69 с., 48 ил., 12 табл., библиогр. список – 30 наим., 1 прил.

Работа посвящена разработке программного комплекса «Аренда апартаментов».

В работе осуществлен обзор и анализ существующих интернет-сервисов, работающих с недвижимостью, имеющих назначение аналогичное реализованному в данной работе программному продукту.

Приведено описание предметной области, на основании которой спроектирована БД и разработана структура сайта.

Третий раздел посвящён реализации приложения, описан пользовательский интерфейс программы и представлены результаты ее тестирования.

Оглавление

ВВЕДЕНИЕ

Разработка клиент-серверных приложений, web-сайтов и прочих сервисов, позволяющих удаленно выполнять операции, экономя при этом время, востребованы и очень важны для бизнеса. Во многих компаниях имеется ресурс, позволяющий часть работ передать автоматизированным средствам, уменьшая количество «брака» связанного с человеческим фактором (к примеру, невнимательностью).

Бизнес старается переходить в онлайн режим. Это означает, что привычные операторы заказа такси заменяются системами онлайн заказа (Яндекс, Uber), банки переходят в онлайн режим работы (Тинькофф, Сбербанк), а некоторые застройщики дают возможность не только забронировать, но и оплатить залог удаленно, без посещения офиса отдела продаж, а в скором времени покупка жилья в интернет магазине будет обычным явлением.

Удаленный режим работы увеличивает вероятность бесперебойной работы компании. Это особенно актуально сейчас, когда в мире «царит» пандемия и необходимо минимизировать контакт клиента и сотрудника. И решения сдачи квартир в аренду в онлайн режиме, не исключение.

На сегодняшний день в России работают сотни агентств недвижимости. Подобное расширение этих учреждений обусловлено востребованностью их услуг. При помощи специализированных риелторских предприятий можно снять или купить: производственные объекты, складские помещения, конторские помещения, земельные участки, частные дома, квартиры, комнаты.

Востребованность риелторских компаний выражается не только широким спектром предоставляемых услуг. Также стоит отметить обширную клиентскую базу данных организаций. Большая часть нынешних агентств недвижимости функционирует на рынке ни один десяток лет. Благодаря

этому их клиентские базы состоят из тысяч потенциальных арендаторов, арендодателей, продавцов и покупателей.

Данная тема актуальна, так как агентствами по недвижимости пользуется огромное количество людей, которые снимают, либо сдают жилье.

К малой части сервисов, в которых можно арендовать недвижимость, относятся:

1) https://www.avito.ru/;
2) https://www.cian.ru/;
3) https://www.booking.com/.

Данные примеры отличаются друг от друга, и имеют разные подходы к предоставлению пользователю информации.

Целью настоящей выпускной квалификационной работы является создание сайта, предназначенного для решения более узкой задачи, а именно для сдачи и бронирования апартаментов.

Проектируемый сайт должен выполнять следующие задачи:

1) обеспечить возможность целевой аудитории более детально ознакомиться с предоставляемыми услугами по аренде жилья;
2) сайт должен быть понятным и удобным в использовании;
3) упростить процедуры аренды помещения;
4) предоставить возможность связываться с покупателями и получать от них обратную связь;
5) работать 24 часа в сутки.

В процессе постановки задачи определены функции, возможность выполнения которых должен предоставлять сайт:

1) выполнять регистрацию и авторизацию пользователей, в статусах либо арендодатель, либо арендатор;
2) предоставлять возможность арендодателям создавать объявление об аренде апартаментов;

3) предоставлять возможность владельцам апартаментов (арендаторам) изменять цену аренды апартаментов;

4) предоставлять возможность арендаторам выполнять бронирование апартаментов, а также отмену своей брони;

5) обеспечивать добавление, поиск и выдачу сведений о владельцах квартир, арендаторах;

6) предоставлять сведения о сдаваемых квартирах, их адресах, владельцах, этажности, площади, количестве комнат, стоимости арендной платы, удобствах.

На основании анализа приведенных выше требований были сформулированы задачи выпускной квалификационной работы:

1) изучение предметной области;

2) формулировка требований и бизнес-процессов при выполнении аренды;

3) определение ролей и функций участников бизнес-процессов;

4) построение информационно-логической модели;

5) проектирование базы данных;

6) разработка структуры сайта;

7) разработка функционала пользователей сайта;

8) написание, отладка и тестирование программного кода, реализующего необходимые функции;

9) публикация сайта в интернете.

Для создания программного обеспечения были выбраны следующие средства реализации:

- Open Server – портативный локальный WAMP/WNMP сервер, имеющий многофункциональную управляющую программу и большой выбор подключаемых компонентов;
- MySQL – сервер баз данных, бесплатно предоставляемый практически всеми хостингами;

- Laravel – бесплатный PHP web-фреймворк с открытым кодом, предназначенный для разработки с использованием архитектурной модели MVC (модель-представление-контроллер);
- Bootstrap – открытый и бесплатный HTML, CSS и JS фреймворк, который используется веб-разработчиками для быстрой вёрстки адаптивных дизайнов сайтов и веб-приложений.

1 ОБЗОР СУЩЕСТВУЮЩИХ ИНТЕРНЕТ-СЕРВИСОВ. ВЫБОР СРЕДСТВ ДЛЯ СОЗДАНИЯ САЙТА. АНАЛИЗ ТРЕБОВАНИЙ К САЙТУ

1.1 Существующие интернет-сервисы аренды недвижимости

1.1.1 Интернет-сервис «Авито»

Авито представляет собой электронный каталог объявлений о товарах, услугах, вакансиях и других предложениях, которые пользователи могут предлагать и искать на Авито с целью заключения сделок [1].

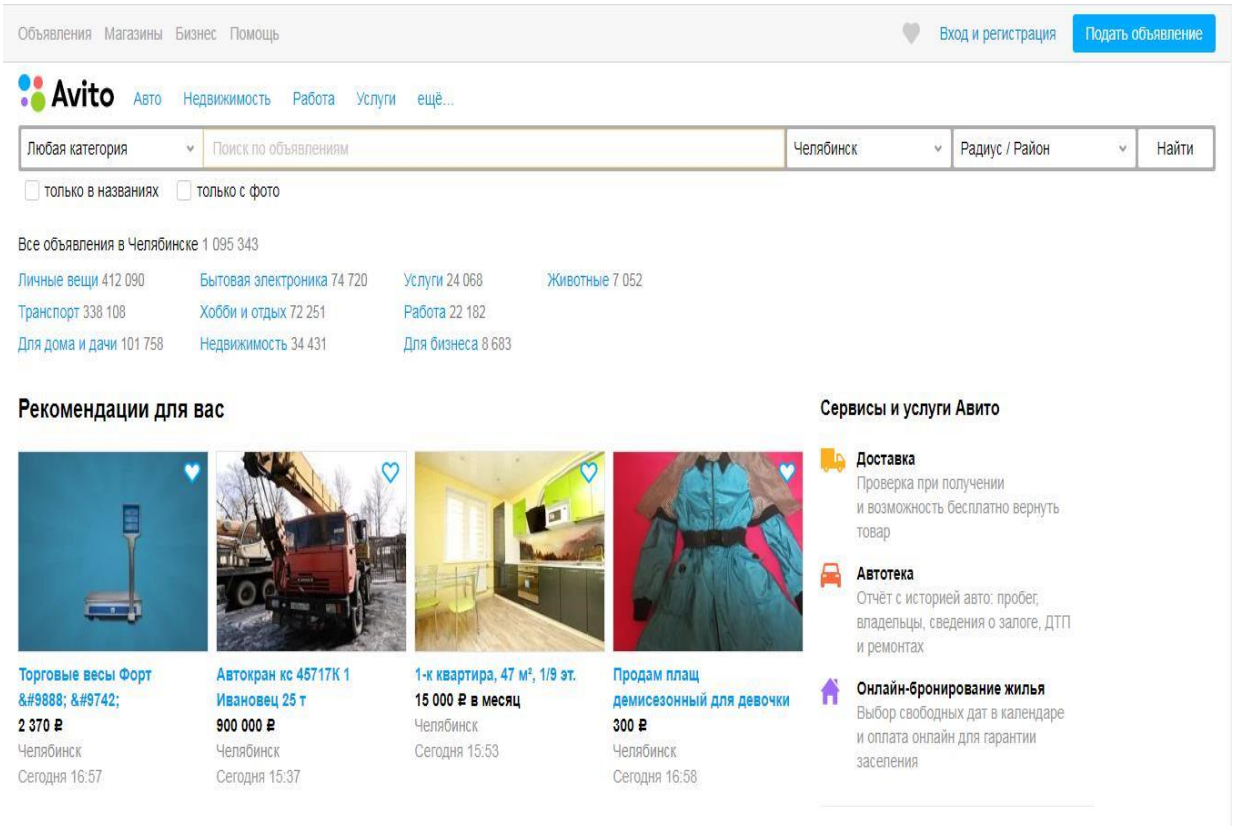

Рисунок 1.1 – Главная страница интерент-сервиса Avito

На рисунке 1.1 в верхнем правом углу предоставлена возможность пользователю войти и зарегистрироваться, а также подать объявление. Строчкой ниже пользователь может просмотреть доступные категории сервиса: «авто», «недвижимость», «работа», «услуги» и «еще».

На сайте есть возможность искать объявления только с фотографией. Также на рисунке 1.1 видно все объявления в определенной местности, в данном случае в городе Челябинск и предложенные рекомендации.

При выборе категории «Недвижимость» откроется окно, представленное на рисунке 1.2 со всеми возможными вариантами апартаментами: квартиры, дома, дачи, комнаты и пр. Для удобства здесь есть возможность посмотреть объявления на карте.

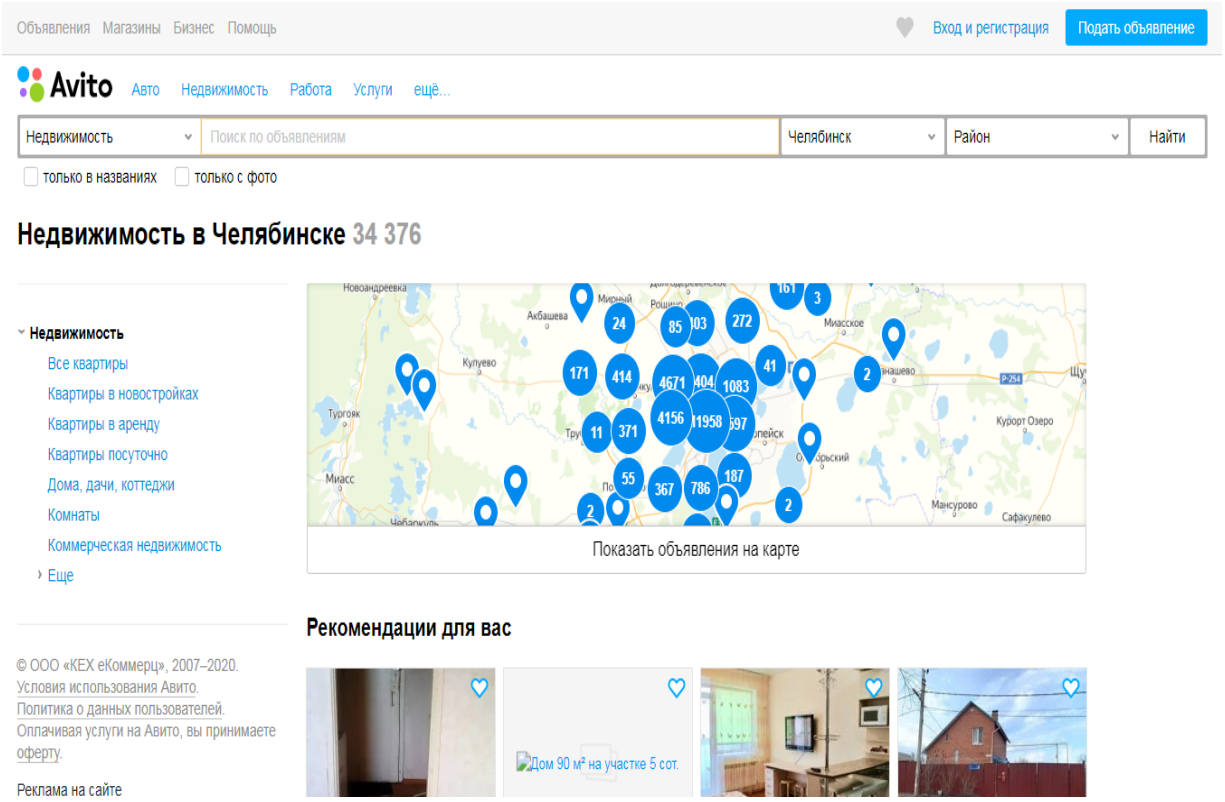

Рисунок 1.2 – категория «Недвижимость» интернет-сервиса Avito

Чтобы подыскать комфортное помещение для аренды нужно ввести желаемые параметры помещения, которые изображены на рисунке 1.3, а именно:

- даты аренды;
- количество комнат;
- арендную плату (руб./месяц или сутки);
- общая площадь (кв. м);
- этаж;
- количество этажей в доме;
- тип дома и арендодателя (частник или агентство).

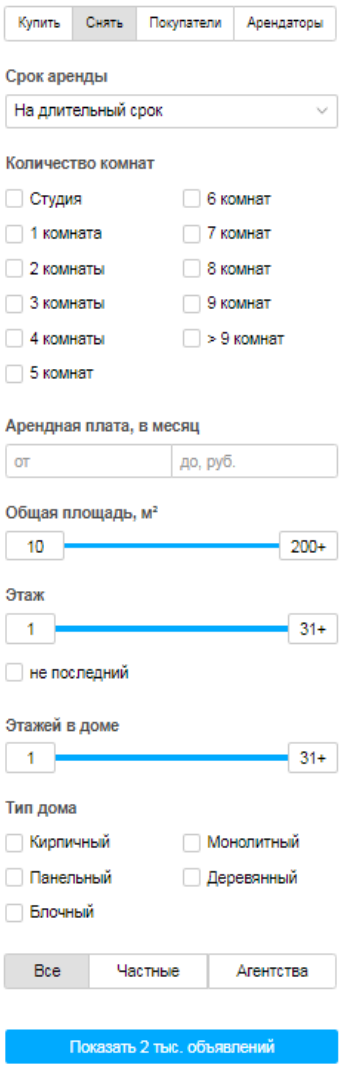

Рисунок 1.3 – Пункты, которые нужно заполнить
для аренды помещения в Avito

На рисунке 1.4 изображен пример объявления о недвижимости. Пользователь может увидеть фотографии квартиры с ее описанием: этаж, количество комнат, площадь, адрес, цену и некую информацию о самом доме, в котором находится квартира, увидеть номер арендодателя и связаться с ним сможет только зарегистрированный пользователь.

Все пользователи (независимо от роли) имеют возможность искать и просматривать объявления на сайте. Пользователи используют размещенную на интернет-сервисе информацию, чтобы заключать сделки на свой страх и риск без прямого или косвенного участия или контроля со стороны интернет-сервиса.

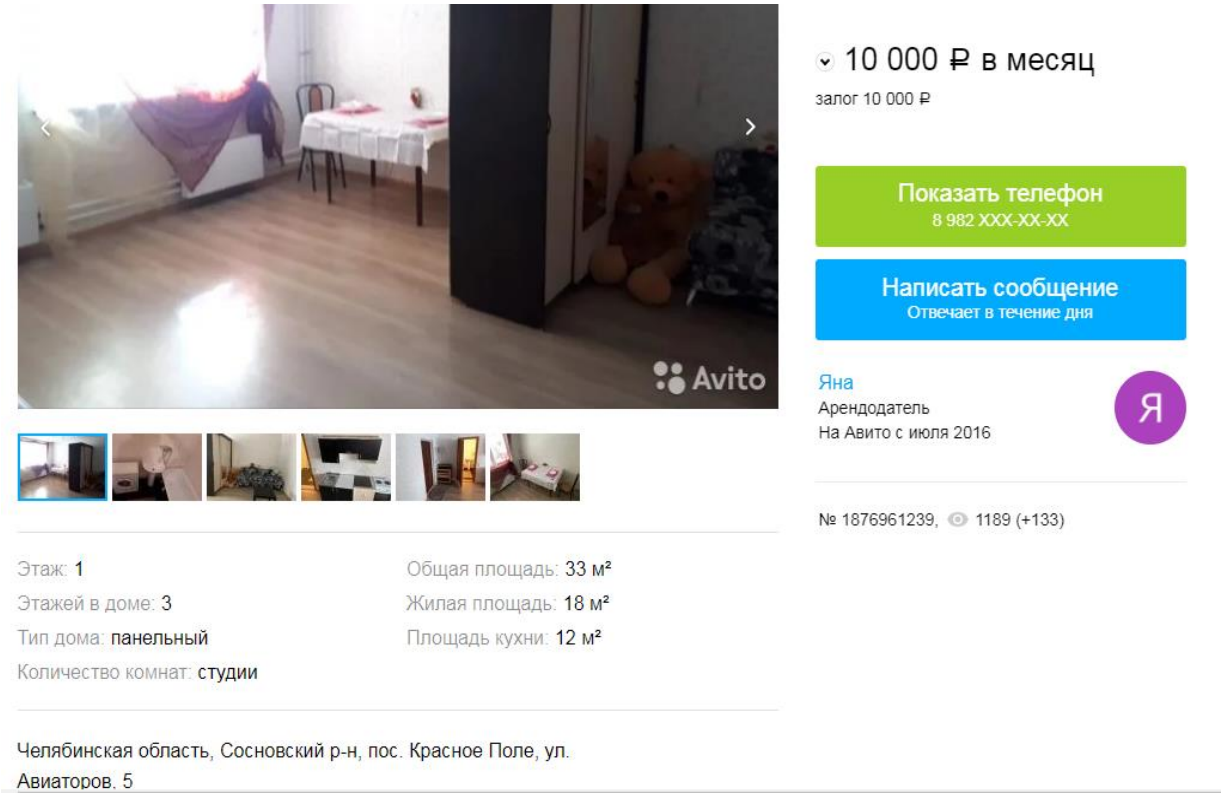

Рисунок 1.4 – пример объявления о сдачи квартиры в Avito

Привлекательными характеристиками сайта являются:

- функциональный, но одновременно простой интерфейс;
- обратная связь и возможность списываться с поставщиками и потребителями услуг;
- огромное число объявлений;
- мобильное приложение;
- большая аудитория.

Имеются некоторые отрицательные моменты:

- не все объявления являются достоверными, следовательно, есть возможность столкнуться с мошенниками;
- сайт является многопрофильным и, как следствие, появляется много ненужной рекламы;
- непродуманная система рейтинга продавцов;

- отсутствует возможность детализации признаков поиска (например: количество комнат, наличие стиральной машины, наличие балкона, приятный вид из окна и т. д.).

1.1.2 Интернет-сервис «Циан»

Cian.ru – крупнейший интернет-сервис для покупателей и арендаторов жилья [2]. Начальная страница приложения изображена на рисунке 1.5. Здесь так же как и в Avito пользователь может войти и зарегистрироваться на сайте и разместить свое объявление [23]. Отличие состоит только в том, что Циан имеет дело исключительно с недвижимостью, что более подходит к нашей теме.

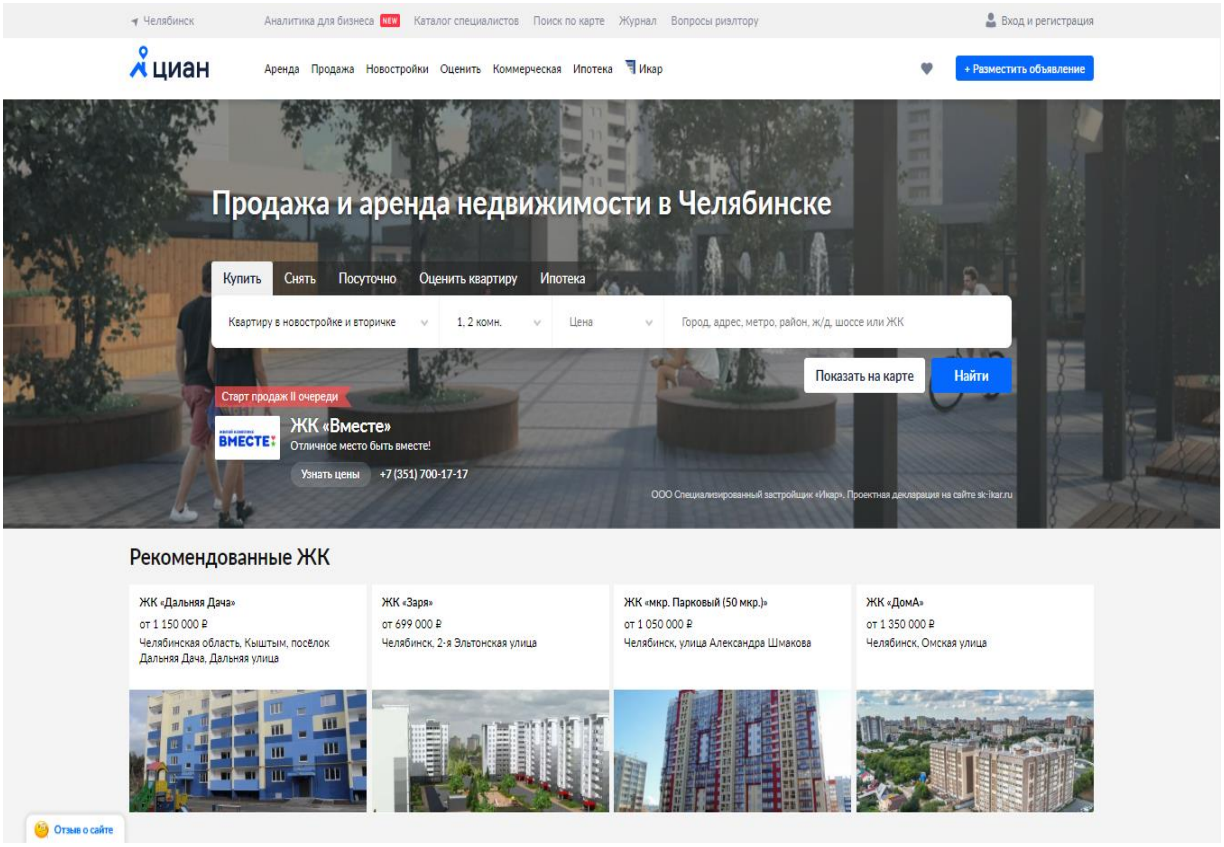

Рисунок 1.5 – главное окно интернет-сервиса Циан

На сайте пользователь может купить или арендовать помещение (длительно или посуточно), соответственно, перед этим посмотреть стоимость помещения, количество комнат и само местоположение этого помещения, также для удобства на сайте есть карта, которой может

воспользоваться пользователь, чтобы на ней найти свой объект недвижимости. Сайт имеет нужную функцию, с помощью которой можно узнать рыночную стоимость и арендную ставку квартиры. Для этого нужно ввести населенный пункт, улицу, дом, № квартиры, количество комнат, площадь и цель оценки.

Признаками, по которым осуществляется отбор подходящих апартаментов являются (рисунок 1.6):

- тип помещения (квартира, комната, дом, офис и пр.);
- количество комнат;
- цена;
- адрес;
- дополнительные параметры (холодильник, кондиционер);
- площадь помещения;
- планировка;
- этаж;
- количество этажей в доме и многое другое.

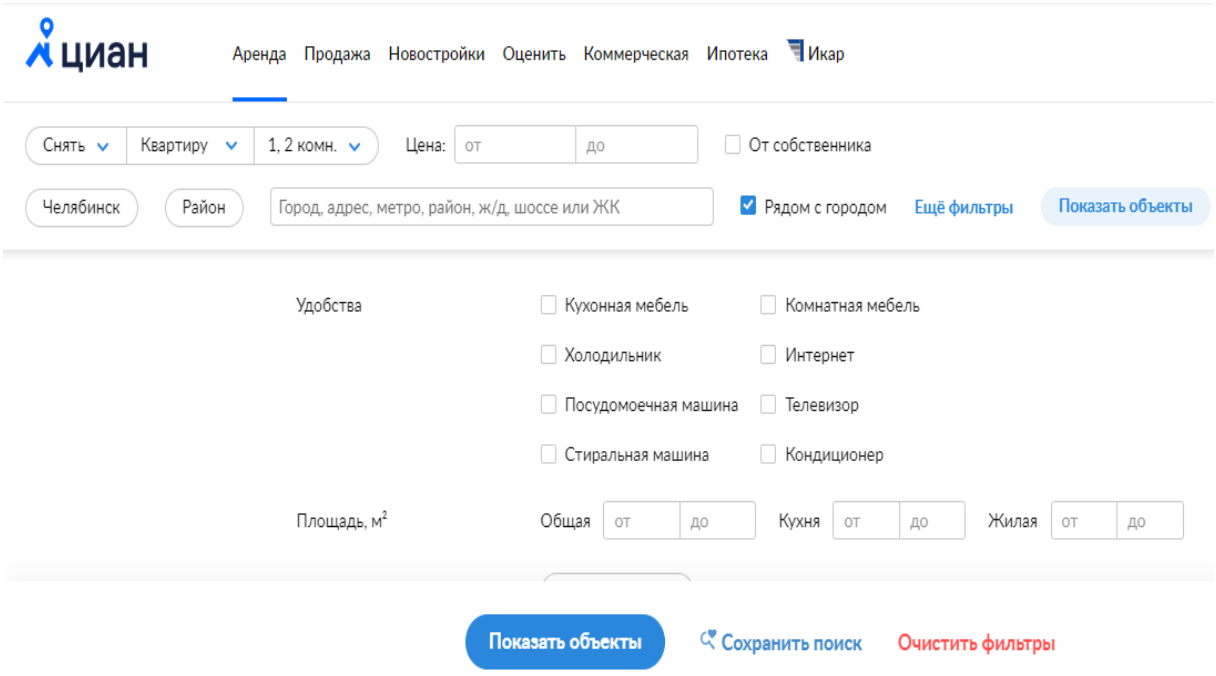

Рисунок 1.6 – Пункты, которые нужно заполнить для аренды помещения в Циан

Ниже приведен рисунок 1.7 с объявлением квартиры.

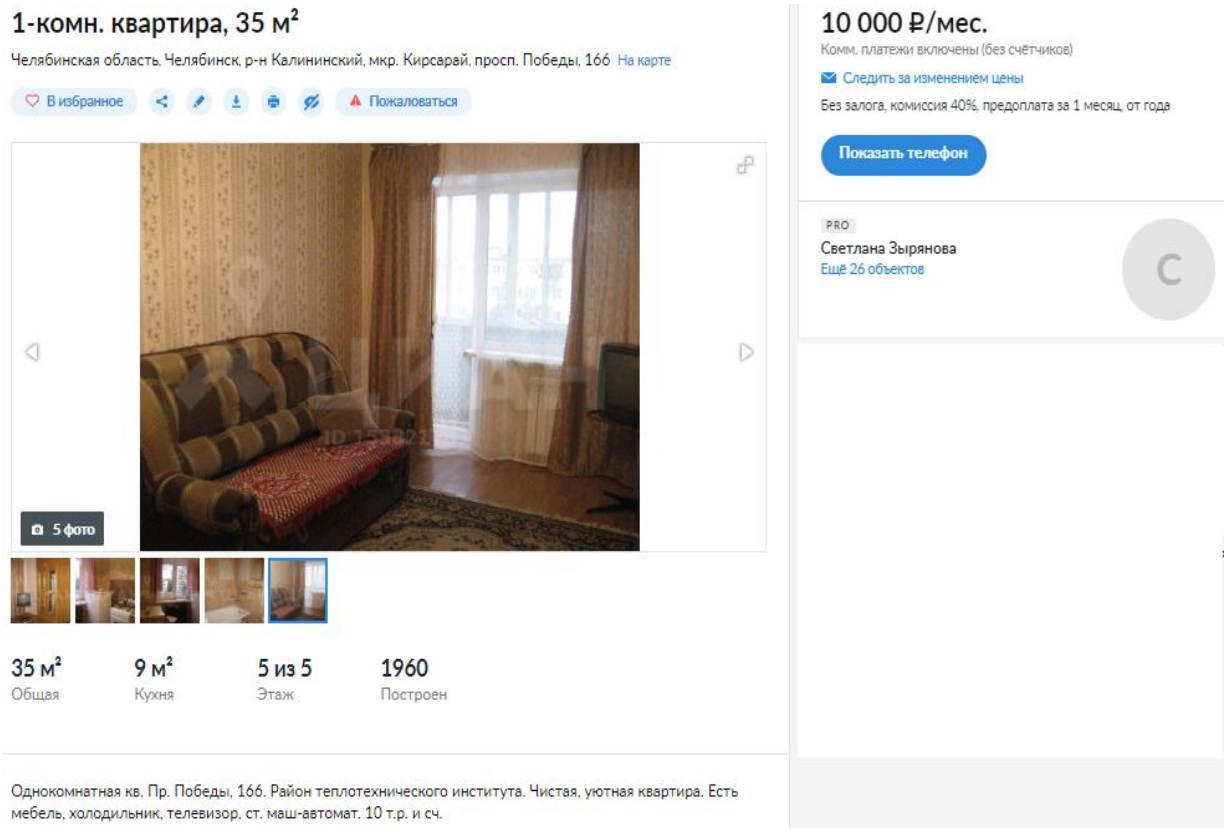

Рисунок 1.7 – Пример объявления

На рисунке 1.7 представлен пример объявления о сдачи квартиры. В объявление входят фотографии и описание квартиры (число комнат, площадь, этаж, полный адрес, удобства и стоимость аренды), также предоставлен номер телефона, по которому можно обратиться к арендодателю.

Анализируя этот сайт можно выделить положительные черты:

- привлекательный и современный дизайн сайта;
- удобная навигация по сайту;
- огромная база недвижимости;
- достаточно большой набор признаков, которые помогают осуществлять отбор;
- возможность найти недвижимость в любой географической точке.

Минусы:

- сайт не несет ответственность за размещенные объявления, следовательно, можно столкнуться с аферистами;
- малое количество объявлений от собственников.

1.1.3 Агентство недвижимости «Компаньон»

«Компаньон» является крупнейшим участником рынка недвижимости [12]. На рисунке 1.8 изображена главная страница сайта компании.

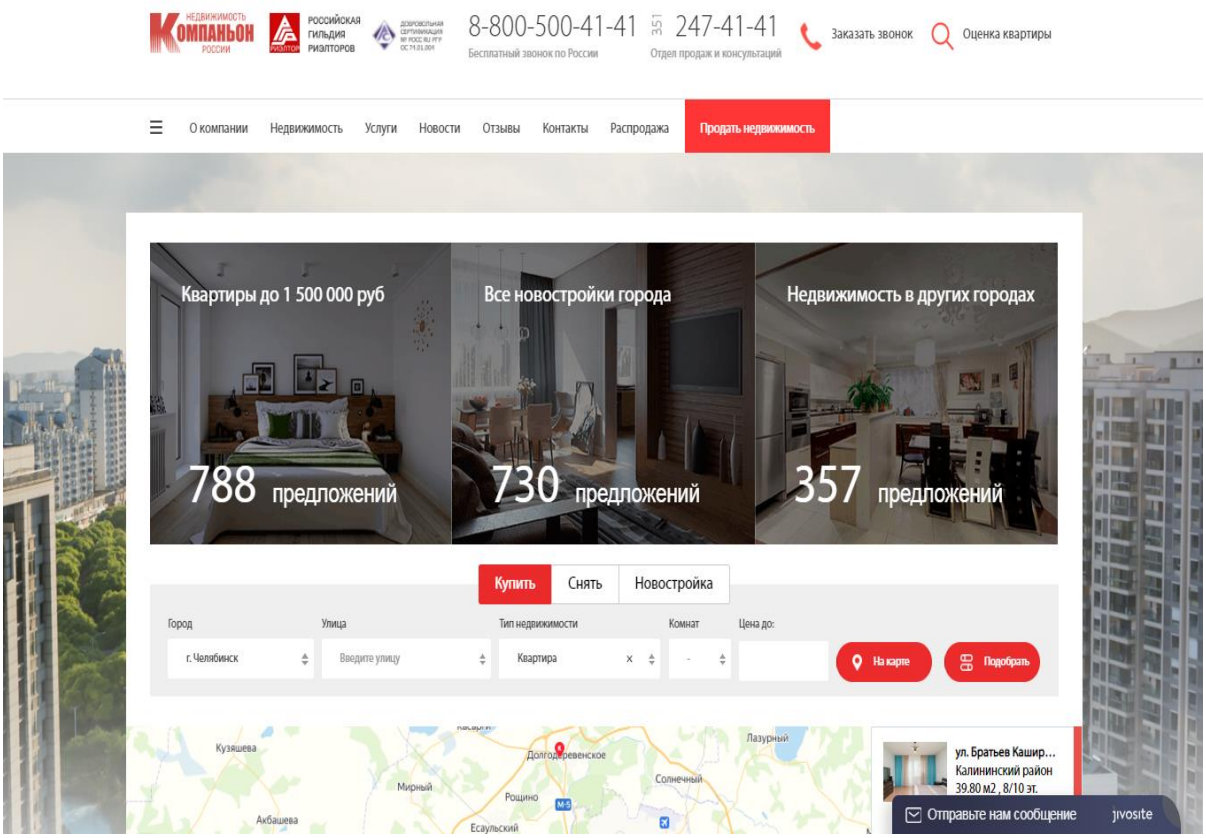

Рисунок 1.8 – Главное окно агентства недвижимости «Компаньон»

В шапке страницы сайта представлен логотип предприятия и контактную информацию. Ниже есть возможность посмотреть информацию о компании, какие услуги она предоставляет; можно увидеть недвижимость и посмотреть отзывы о сайте.

Чтобы снять апартаменты, необходимо заполнить информацию, изображенную на рисунке 1.9:

- адрес;
- тип помещения;

- цена;
- количество комнат;
- тип дома;
- общая площадь;
- планировку;
- этаж;
- количество этажей.

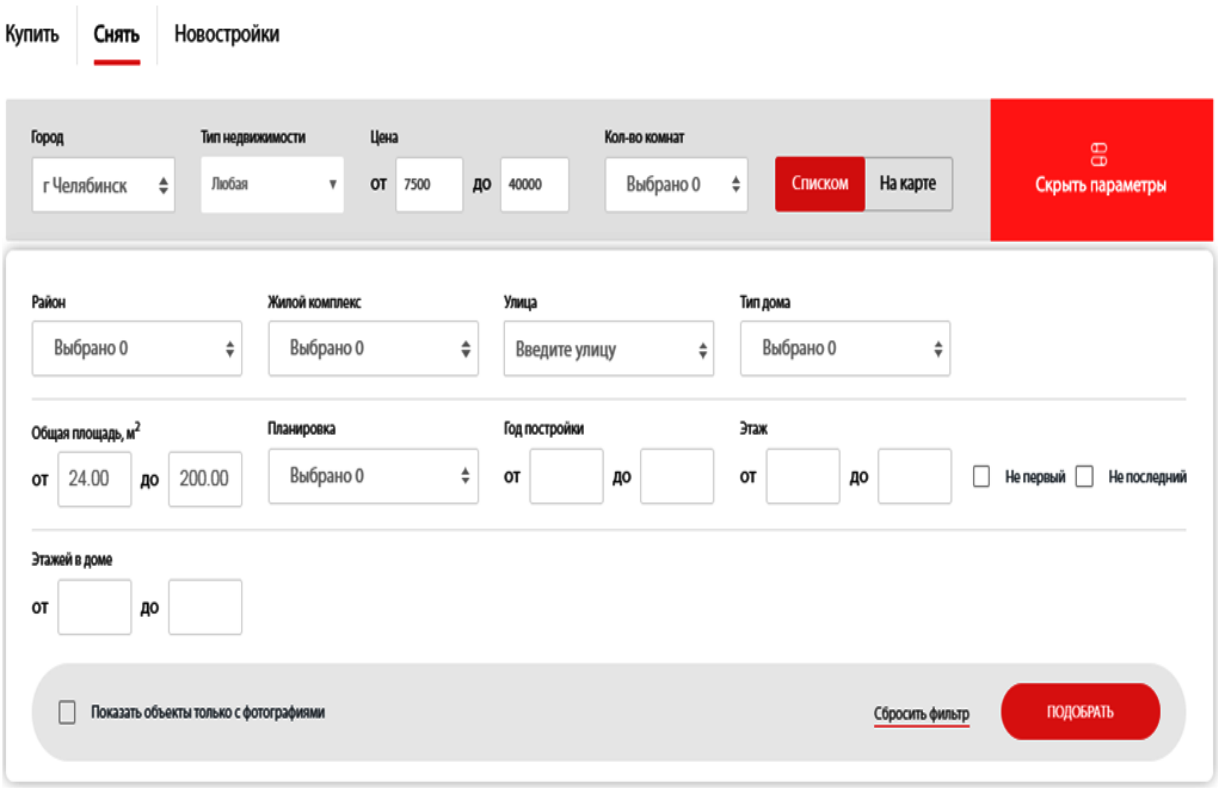

Рисунок 1.9 – Пункты, которые нужно заполнить для аренды помещения в Компаньоне

К достоинствам сайта стоит отнести:

- приятный глазу, дружественный дизайн сайта;
- возможность увидеть помещение на карте;
- реализована возможность обратной связи с агентством.

К недостаткам относится:

- мало фильтров, нет возможности выбрать дополнительные удобства;
- наличие недостоверной информации, ненадежность сайта.

1.1.4 Система интернет-бронирования отелей «Booking.com»

Booking.com – это система интернет-бронирования отелей и апартаментов, основана в Амстердаме в 1996 году [4]. Главная страница этого сайта изображена на рисунке 1.10.

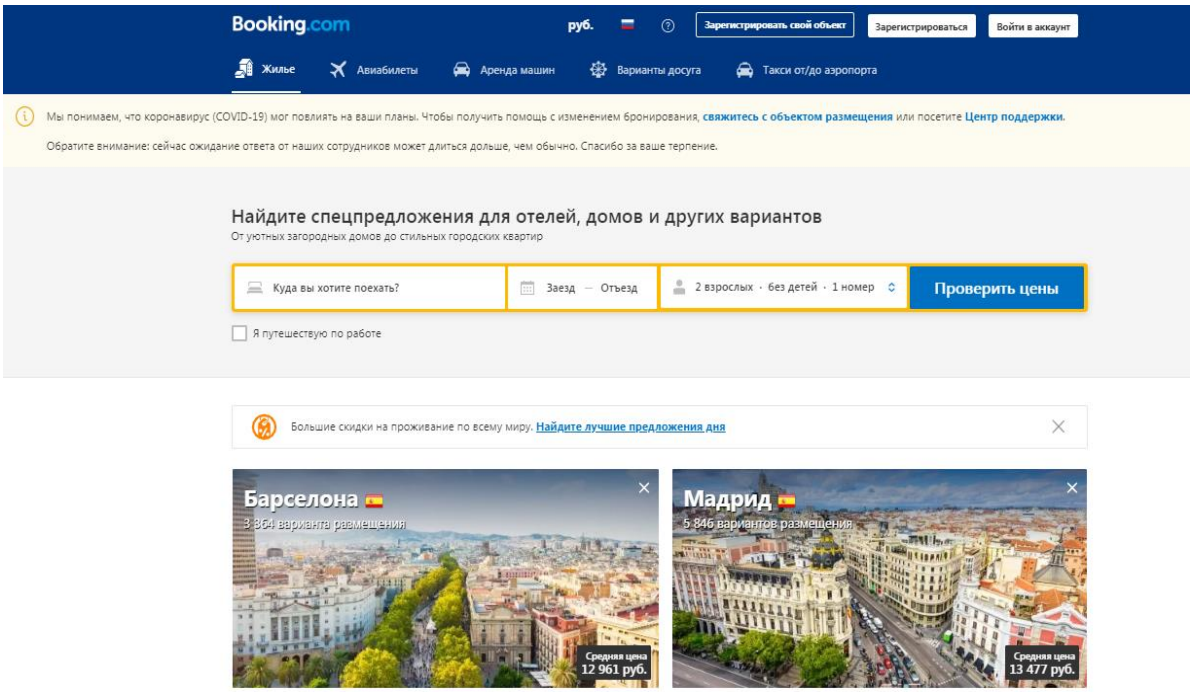

Рисунок 1.10 – Главная страница интернет-сервиса Booking.com

Головная страница предоставляет возможность выбрать валюту и язык, а также зарегистрироваться и войти в аккаунт под своей учетной записью.

Сервис занимается не только недвижимостью, поэтому также можно купить авиабилеты, арендовать транспорт и заказать такси от/до аэропорта.

Для того, чтобы арендовать отель нужно заполнить следующие пункты (рисунок 1.11):

- город;
- дату заезда;
- дату отъезда;
- количество людей;
- количество номеров.

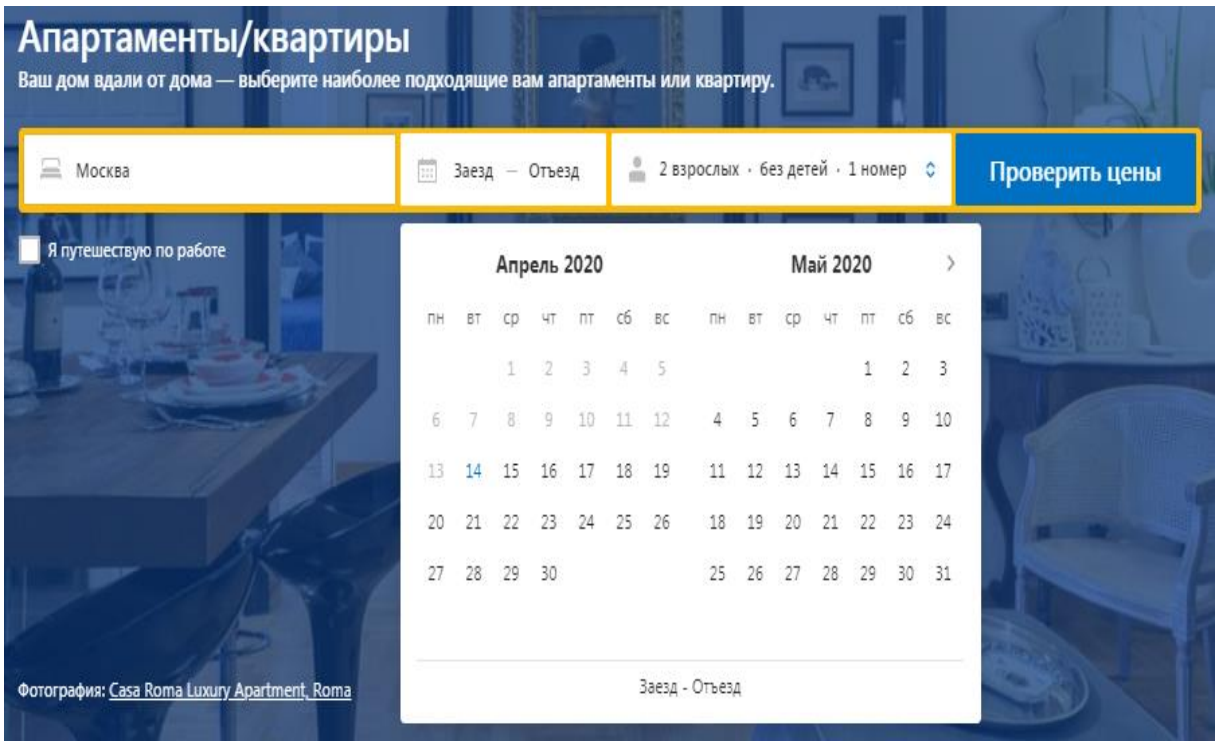

Рисунок 1.11 – Процесс аренды апартаментов в Booking.com

После нажатия кнопки «Проверить цены» предъявляются подходящие варианты отелей (рисунок 1.12).

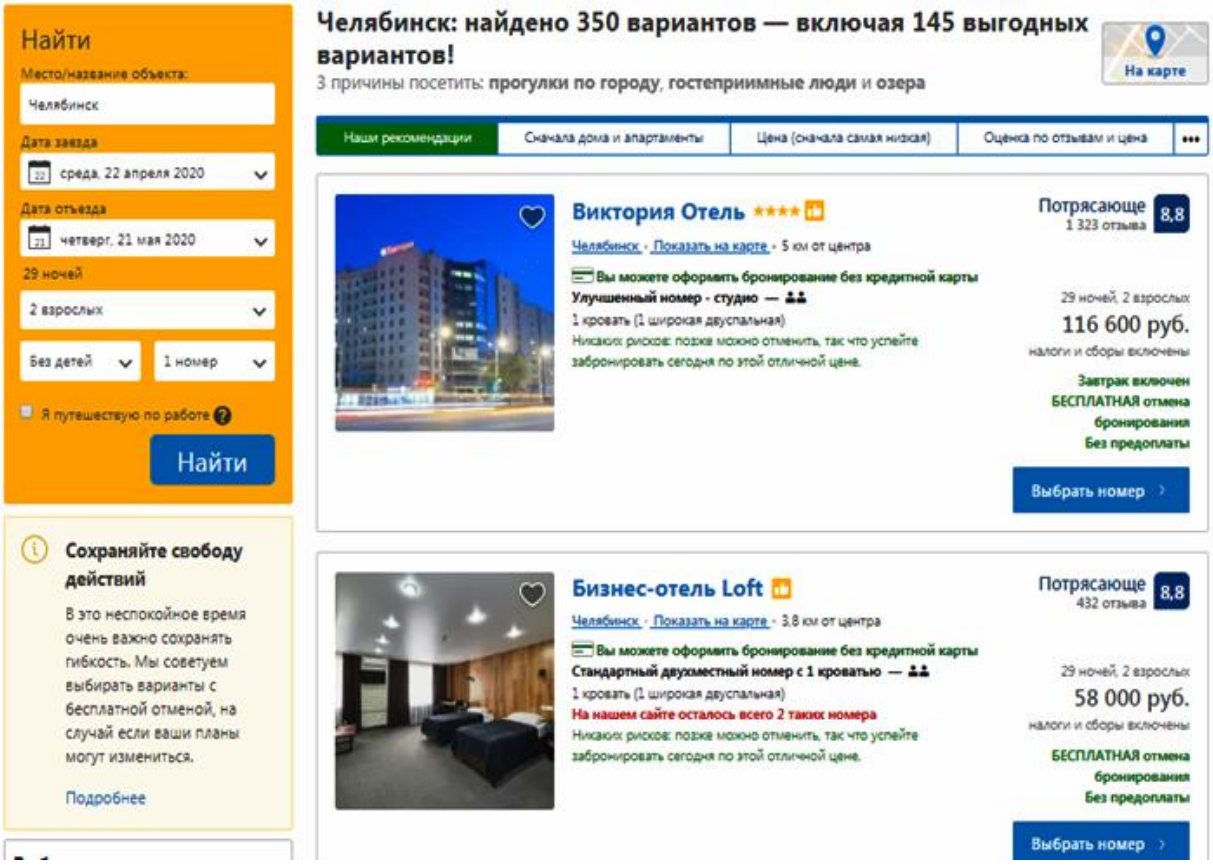

Рисунок 1.12 – Варианты отелей в Booking.com

После этого можно корректировать выбор жилья с помощью дополнительных фильтров изображенных на рисунке 1.13. Можно найти отель исходя из своего бюджета, количества звезд отеля, расстояния от отеля до центра города. Также можно использовать фильтры: оценка по отзывам, тип кровати, удобства в номере и многое другое.

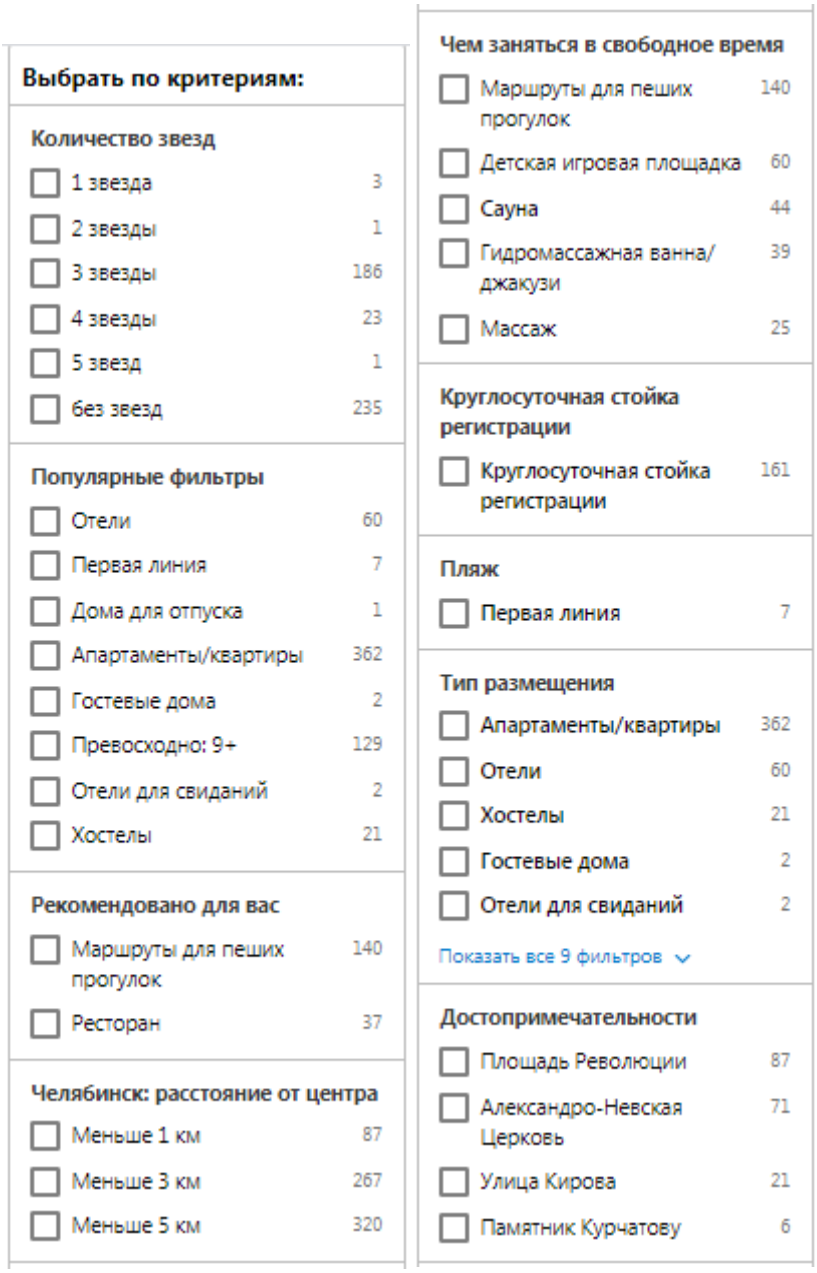

Рисунок 1.13 – Часть фильтров в Booking.com

После заполнения всех пунктов можно уже окончательно найти подходящее место для проживания. Пример отеля, изображаемого на странице этого сайта, приведен на рисунке 1.14. Далее стоит лишь нажать кнопку «Забронировать», ввести свои данные и снять отель.

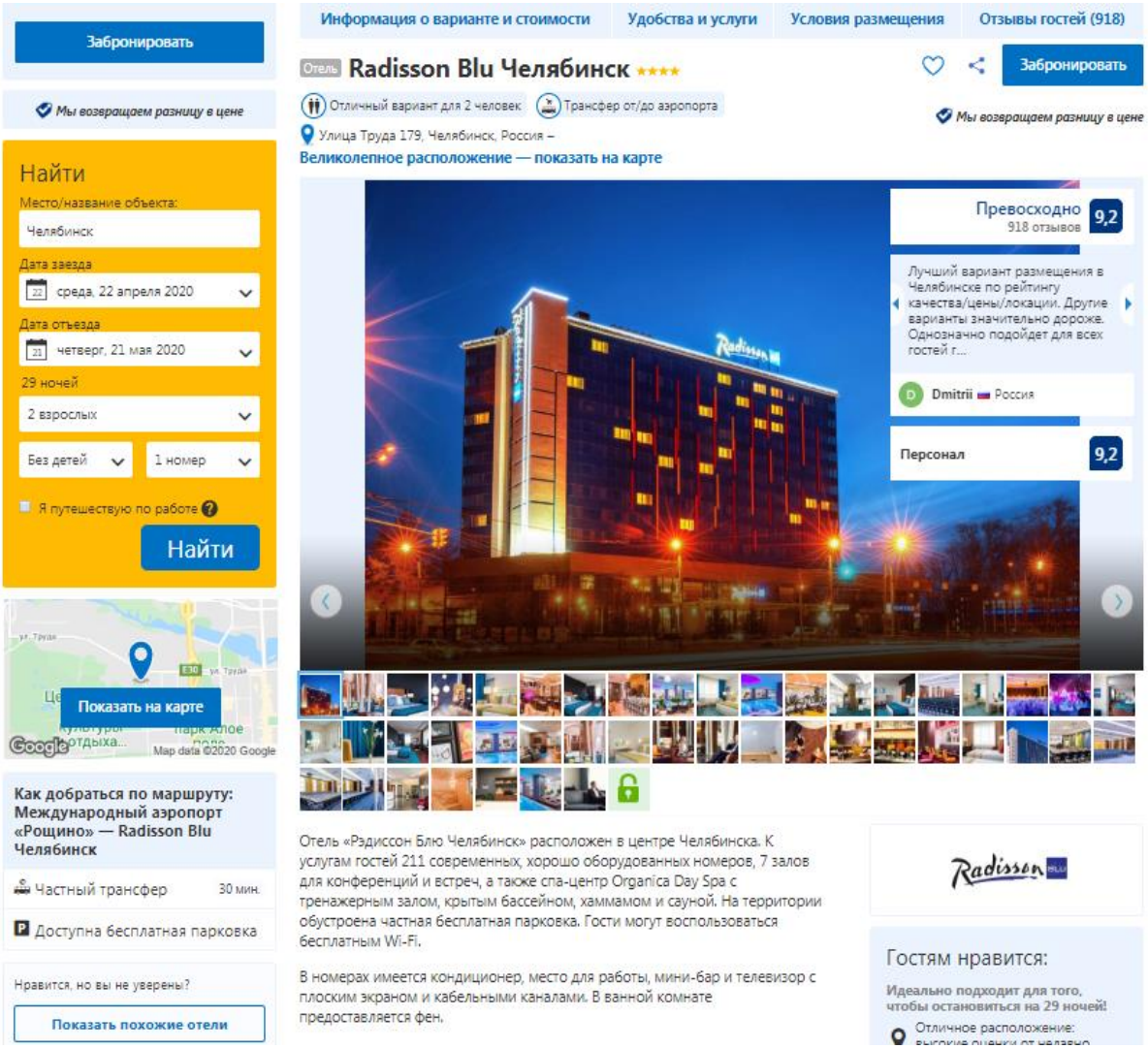

Рисунок 1.14 – Пример отеля в Booking.com

К положительным чертам сайта относятся:

- простота бронирования отелей;
- можно просмотреть отзывы об отеле на той же странице;
- существует возможность посмотреть расположение отеля на карте;
- много фильтров для поиска наиболее подходящего помещения;
- есть функция бесплатной отмены бронирования.

Также минусы:

- Booking.com не несёт никакой ответственности за размещённые объявления;
- слабая техподдержка.

1.1.5 Вывод по интернет-сервисам

Рассмотрев все вышеперечисленные интернет-сервисы, следует изобразить обобщающую таблицу 1.1, включающую в себя недостатки и достоинства данных аналогов.

Таблица 1.1 – Таблица аналогов

Критерий	Авито	Циан	Компаньон	Booking.com
Привлекательный интерфейс	+	+	+	+
Большой выбор фильтров для поиска	–	+	–	+
Мобильная версия	+	+	–	+
Возможность поиска по карте	+	+	+	+
Рассылка новостей	–	+	–	+

1.2 Бизнес-процесс

Основным бизнес-направлением сайта является сдача жилья под аренду. Основной задачей для достижения цели агентства недвижимости является подбор высококвалифицированного кадрового персонала, готового обеспечить должный уровень оказания услуг.

Кадровый состав современного агентства недвижимости должен включать в себя:

- директора агентства недвижимости;
- администратора;
- арендодателей;

- вспомогательный управляющий персонал (юрист).

Руководитель предприятия (директор), администратор и юридический отдел не участвуют в непосредственном контакте с арендатором, осуществляя общие мероприятия, направленные на обеспечение жизнедеятельности предприятия.

Непосредственный контакт клиента, нуждающегося в снятии жилья, начинается со встречи с арендодателем.

На условии обязательного соблюдения принципов делового этикета (вежливость, искренняя заинтересованность в клиенте и т. п.) сотрудник в лице арендодателя осуществляет подбор жилья с учетом требований, заявленных арендатором. После выяснения всех требований, арендодатель консультирует клиента об апартаменте, рассказывает об удобствах, включенных в этот апартамент, и оформляет договор о сдаче жилья.

1.3 Формулирование требований

Основа разработки это понимание требований системы. С целью определения требований проведен ряд интервью среди лиц, имеющих опыт со сдачей и арендой жилья для сбора первичной информации. В результате этого, с учетом собственного опыта съема недвижимости, были сформулированы требования.

Основные требования к дизайну:

- сайт должен быть узнаваем, а, следовательно, иметь свое название и логотип;
- корректно отображаться на экранах мобильных устройств, планшетах, мониторах;
- основные функции должны быть легко доступны;
- от пользователя должно требоваться минимум усилий для бронирования;

- сайт должен быть оформлен в едином стиле (единообразные шрифты, расположение блоков, работа модальных окон);
- минимум рекламных сообщений.

Основные требования к функционалу:

- возможность регистрации и авторизации пользователей, имеющих различную роль;
- личный кабинет пользователя.

Основные требования к функционалу арендодателя:

- возможность добавления, редактирования, удаления сдаваемых апартаментов;
- отслеживание сроков аренды апартаментов;
- возможность отмены брони помещения.

Основные требования к функционалу арендатора:

- возможность бронирования и отмены бронирования апартаментов;
- отслеживание своей истории бронирований;
- возможность просмотра апартаментов без авторизации на сайте.

Из вышеописанного следует, что для разных ролей пользователей должны быть доступны разные функции, так как интересы этих лиц различны.

Возможности сайта должны быть интуитивно понятны для всех категорий пользователей. К примеру, при использовании иконок, стоит использовать те «стандарты», к которым привыкли пользователи.

1.4 Средства реализации

Проанализировав поставленные задачи и требования, полученные опросом, подобраны инструменты реализации, позволяющие реализовать поставленные задачи, а так же заложить возможность динамического развития проекта в дальнейшем, так как использование фреймворка ограничивает разработку менее, чем конструкторы сайтов.

Выполнение части функционала будет на стороне сервера (обработки событий на PHP), а часть логики выполняться на стороне клиента (отображение модальных окон, некоторые анимации).

Код на стороне клиента использует библиотеку JQuery, входящую в состав пакета Bootstrap. Для обработки событий, не требующих перезагрузки страницы, использована входящая в эту библиотеку технология AJAX.

Выполнение кода на стороне сервера, подразумевает использование языка программирования PHP 7.1.

HTML – стандартизорованный язык разметки документов во Всемирной паутине. Язык HTML интерпретируется браузерами; полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства.

CSS – язык таблиц стилей, который позволяет прикреплять стиль (например, шрифты и цвет) к структурированным документам (например, документам HTML и приложениям XML). Обычно CSS-стили используются для создания и изменения стиля элементов веб-страниц и пользовательских интерфейсов, написанных на языках HTML и XHTML.

В данном проекте используется Bootstrap, содержащий большое количество методов предоставления информации пользователю.

В качестве сервера базы данных используется MySQL, предоставляемая практически всеми хостингами.

В роли сервера будет использован Nginx.

Nginx – это бесплатный веб- и почтовый прокси-сервер с непоточной (асинхронной) архитектурой и открытым кодом.

Apache - это программное обеспечение с открытым исходным кодом, разработанное и поддерживаемое открытым сообществом разработчиков и работающее в самых разных операционных системах.

Сравнение Apache и Ngnix (таблица 1.2) [23].

Таблица 1.2 – Сравнение серверов Apache и Ngnix

Особенность	Apache	Ngnix
Простота	Легко разрабатывать и внедрять новые функции благодаря своей модели «одно соединение на весь процесс»	Сложный в разработке, поскольку он имеет сложную архитектуру для одновременной обработки нескольких соединений
Производительность (Статический контент)	Медленно в отображении статического контента	В 2,5 раза быстрее, чем Apache
Производительность (Динамический контент)	Отличная производительность для динамического контента	
Поддержка ОС	Поддерживает все ОС – Unix, а также Windows	
Безопасность	Это безопасный веб-сервер. Понимание и настройка функций безопасности важны	

Можно сделать вывод, что оба веб-сервера мощны, гибки и полнофункциональны, но в данном случае был сделан выбор в пользу Ngnix, потому что в данной работе нужен более производительный и скоростной сервер.

Фреймворк Laravel способен достаточно просто и эффективно работать с ранееописанными инструментами. Для старта работы с фреймворком необходимы знания PHP, MySQL. Они подразумевают наличие навыков работы с HTML и CSS, JavaScript.

Laravel – это набор готовых функций и компонентов, позволяющие ускорять разработку [27]. Инструкция для установки и настройки фреймворка, в частности, имеется на сайте https://laravel.com.

Фреймворк работает по MVC модели (рисунок 1.15). Model-View-Controller – схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель,

представление и контроллер [28]. Это и позволит облегчить дальнейшее развитие сайта.

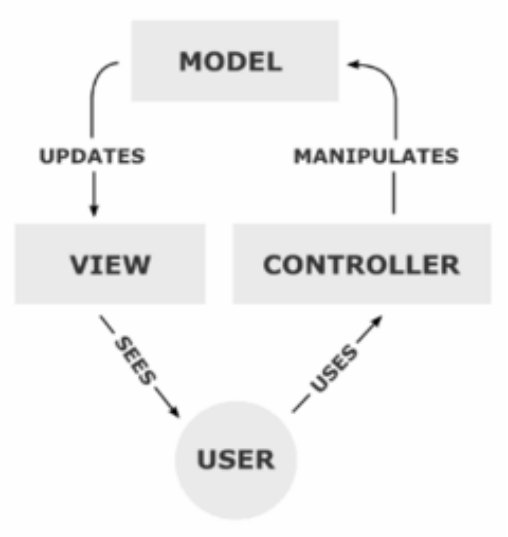

Рисунок 1.15 – Схема MVC модели

Основная цель применения этой концепции состоит в отделении бизнес-логики (модели) от её визуализации (представления, вида). За счёт такого разделения повышается возможность повторного использования кода. Наиболее полезно применение данной концепции в тех случаях, когда пользователь должен видеть те же самые данные одновременно в различных контекстах и/или с различных точек зрения.

К одной модели можно присоединить несколько форм визуального представления информации, при этом, не затрагивая реализацию модели. Например, некоторые данные могут быть одновременно представлены в виде электронной таблицы, гистограммы и круговой диаграммы.

Не затрагивая реализацию форм представления, можно изменять реакции на действия пользователя, например, обработку событий мыши или клавиатуры – для этого достаточно использовать другой контроллер.

Разработчики специализируются только в одной из областей: либо разрабатывают графический интерфейс, либо модель MVC позволяет отделить разработку бизнес-логики от внешнего представления данных.

Поскольку MVC не имеет строгой реализации, то реализован он может быть по-разному. Нет общепринятого определения, где должна располагаться

бизнес-логика. Она может находиться как в контроллере, так и в модели. В последнем случае, модель будет содержать все бизнес-объекты со всеми данными и функциями.

Модель предоставляет данные и методы работы с ними: запросы в базу данных, проверку на корректность. Модель не зависит от представления (не знает способа отображения данных) и контроллера (не имеет точек взаимодействия с пользователем) просто предоставляя доступ к данным и управлению ими.

Модель строится таким образом, чтобы отвечать на запросы, изменяя своё состояние, при этом может быть встроено уведомление «наблюдателей».

Представление отвечает за получение необходимых данных из модели и отправляет их пользователю. Представление не обрабатывает введённые данные пользователя.

Контроллер обеспечивает «связь» между пользователем и системой. Контролирует и направляет данные от пользователя к системе и наоборот. Использует модель и представление для реализации необходимого действия.

1.5 Выводы

В ходе работы были выявлены требования, предъявляемые к программному продукту, сформулирован бизнес-процесс.

В результате анализа имеющихся сервисов выявлено, что реализации ресурсов для предоставления аренды помещений много, при этом каждый демонстрирует свой стиль в оформлении сайта, а также в функциональности, тем самым предоставляя удобство в использовании для каждого отдельного сегмента пользователей.

Выбранные средства реализации позволяют выполнить поставленные задачи, а та же самая технология позволит продолжить сопровождение и расширение функций.

2 ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ ДЛЯ WEB-САЙТА. ОПРЕДЕЛЕНИЕ РОЛЕЙ

2.1 Роли пользователей

Были введены следующие роли пользователей:

Гость – обычный посетитель сайта, который пришел посмотреть апартаменты. Имеет возможность просматривать информацию о предприятии и апартаментах. Ему не требуется быть зарегистрированным, пока он не захочет арендовать помещение.

Арендодатель – владелец жилой недвижимости, предоставляющий ее во временное пользование другим пользователям. Обладает всеми свойствами простого пользователя, но обязательно должен быть зарегистрирован. Добавляет свои апартаменты на сайт и консультирует арендатора.

Арендатор – лицо, которое заключает с собственником жилого имущества договор аренды (временного пользования). Он обязательно должен быть зарегистрирован, если ему нужно будет арендовать помещение.

Модератор – человек, отвечающий за функционирование сайта, на нем лежит ответственность за выложенные объявления, за выдачу ролей пользователям.

2.2 Модель данных

Для формирования концептуальной модели и построения базы данных необходимо провести идентификацию объектов сущности базы данных.

В дальнейшем понадобятся некоторые определения.

Сущность – представление (абстракция) реально существующего объекта, процесса или явления. Наименование сущности должно быть уникально во всей модели. Экземпляр сущности – конкретный объект из этого набора. Например, типом сущности может быть «Апартаменты», а экземпляром – «Информация о Апартаментах».

Например: сущность «Апартаменты» определяет всю информацию о квартирах вообще. Конкретная Квартира 1 является экземпляром сущности «Квартира», а совокупность всех квартир составляет тип сущности.

Атрибут – свойство сущности (объекта). Его имя должно быть уникально в рамках одной сущности. Экземпляр атрибута – конкретное значение свойства.

Связь позволяет моделировать отношения между объектами предметной области. Наименование связи должно быть уникально во всей модели.

Основой для построения логической структуры является концептуальное моделирование, т.е. переход от описания реального мира к модели этого мира. Синтез концептуальной модели производится при помощи ряда методик.

Одной из наиболее популярных семантических моделей данных является модель «Сущность-Связь» или «Объект-Отношение».

На рисунке 2.1 изображена ER-диаграмма будущей базы данных.

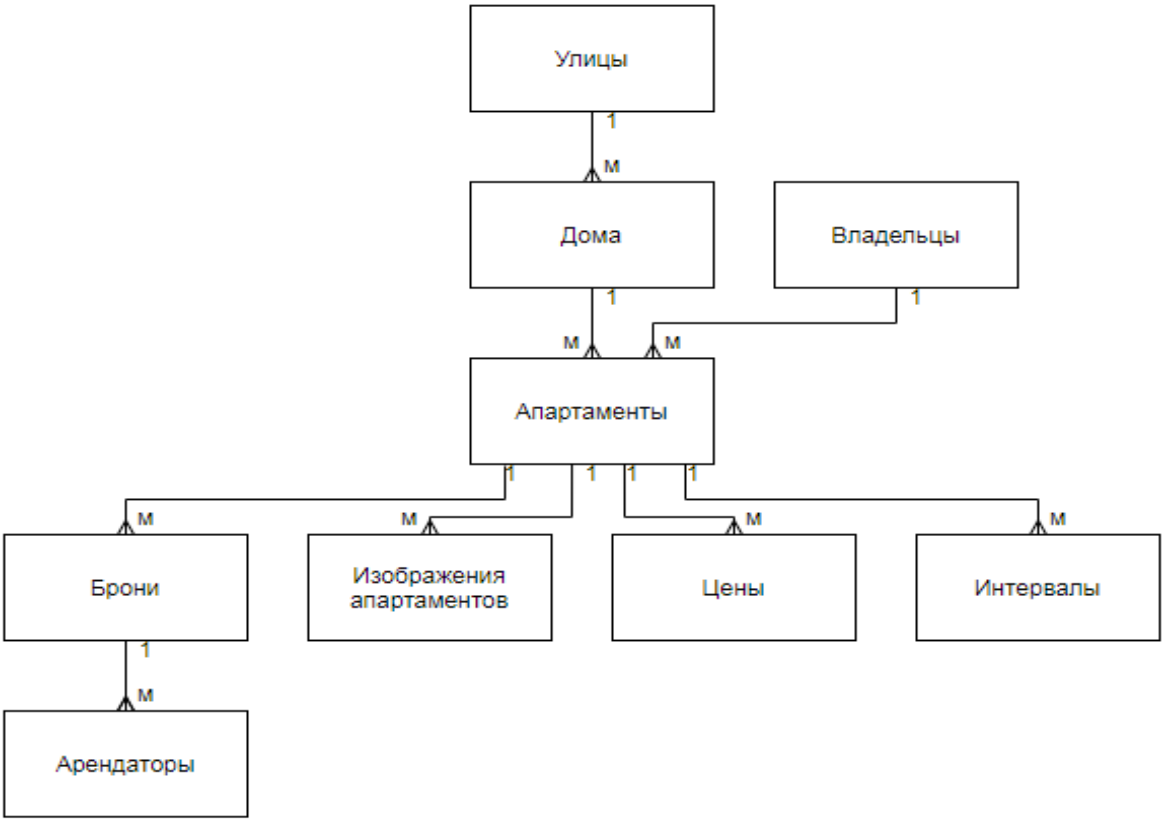

Рисунок 2.1 – ER-диаграмма БД

Перечислим список объектов (сущностей), их атрибуты и связи для предметной области программы «Аренда апартаментов»:

1) «Улицы»:
   - название улицы;
2) «Дома»:
   - улица;
   - номер дома;
   - буква номера дома;
   - название файла картинки дома;
3) «Арендаторы»:
   - ФИО арендатора;
   - эл. почта;
   - логин;
   - пароль;
   - телефон;
4) «Брони»:
   - апартаменты;
   - дата начала брони;
   - дата окончания брони;
   - арендатор;
5) «Владельцы (арендодатели)»:
   - ФИО;
   - эл. почта;
   - логин;
   - пароль;
   - телефон;
6) «Цены»:
   - апартаменты;
   - цена;

- дата начала цены;

7) «Изображения апартаментов»:
   - апартаменты;
   - название файла апартаментов;
   - порядковый номер сортировки при отображении;
8) «Апартаменты»:
   - дом;
   - владелец дома;
   - номер апартаментов;
   - буква апартаментов;
   - количество комнат;
   - площадь апартаментов;
   - площадь кухни;
   - время въезда в апартаменты;
   - время выезда из апартаментов;
   - предельная дата отмены брони для апартаментов;
   - текст объявления;
   - количество просмотров;
   - количество спальных мест;
   - признак «удалено»;
   - признак «активировано»;
9) «Интервалы»:
   - номер апартамента;
   - дата начала;
   - дата окончания.

Разработана схема базы данных, содержащая таблицы:

- Улицы – street;
- Дома – house;
- Арендаторы – renters;

- Брони – booking;
- Владельцы – owners;
- Цены – price;
- Изображения апартаментов – aparts_images;
- Апартаменты – aparts;
- Интервалы – intervals.

В результате анализа предметной области и разработки концептуальной схемы разработана база данных, схема которой изображена на рисунке 2.2.

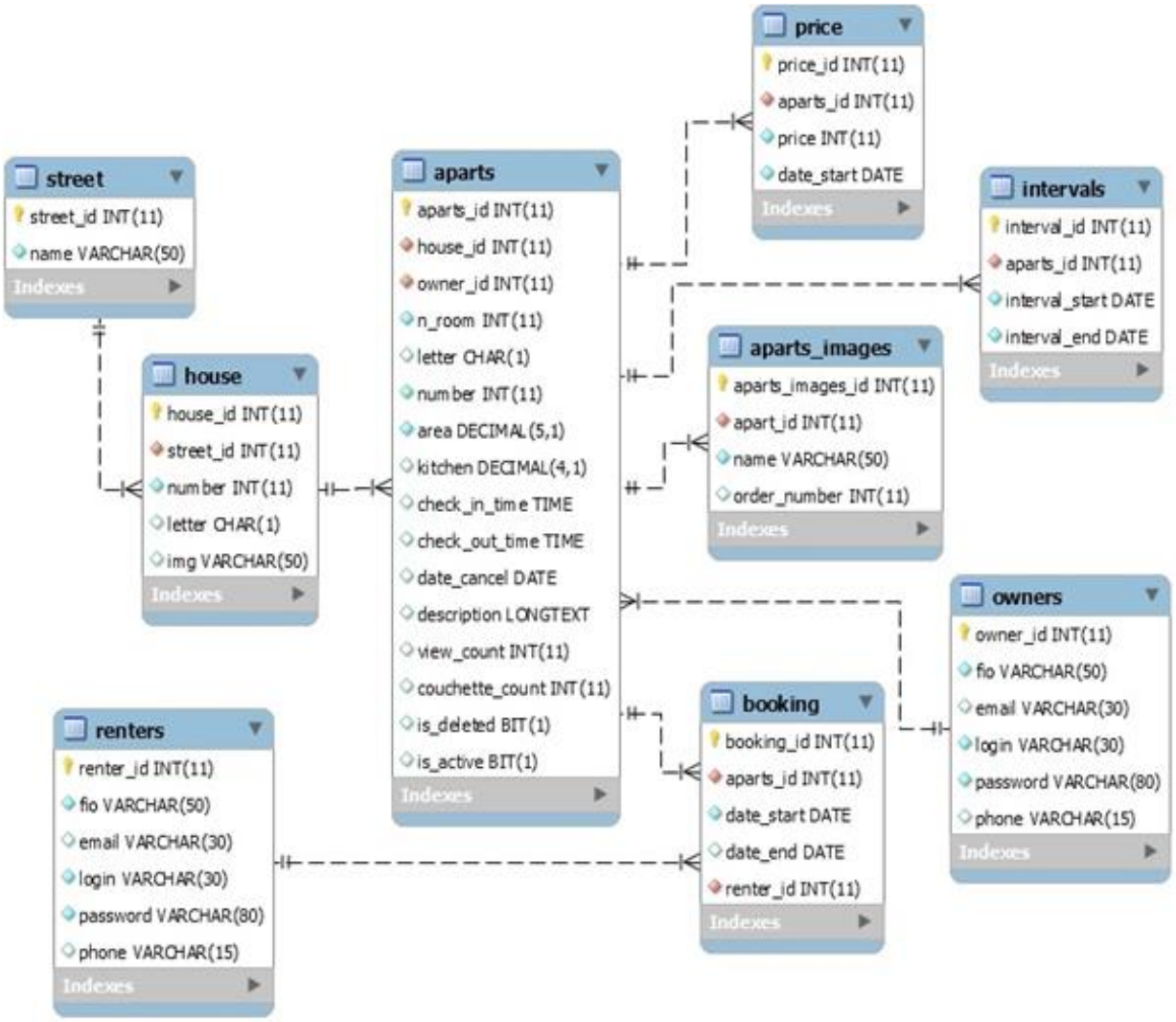

Рисунок 2.2 – Схема базы данных

2.3 Описание базы данных

В таблице 2.1 представлен список названий улиц.

Таблица 2.1 – Таблица street (улицы)

Имя поля	Тип данных	Примечание
street_id	int	Идентификатор улицы, primary key, not null, auto_increment
Name	varchar(50)	Название улицы, уникальное, not null

В таблице 2.2 содержится список домов, в которых находятся апартаменты.

Картинка в данной таблице хранит сгенерированное название файла и расширение. Путь до картинки будет вычисляться по относительному адресу. Все изображения, требуемые для работы сайта, хранятся в отдельной папке, а в базе данных хранятся только ссылки на эти файлы.

Таблица 2.2 – Таблица house (дома)

Имя поля	Тип данных	Примечание
house_id	int	Идентификатор дома, primary key, not null, auto_increment
street_id	int	Идентификатор улицы, на которой находится дом, not null
number	int	Номер дома, not null
letter	char(1)	Буква номера дома, null, например 123а
Img	varchar(50)	Название файла картинки дома, null

В таблице 2.3 содержится список зарегистрированных арендаторов.

Таблица 2.3 – Таблица renters (арендаторы)

Имя поля	Тип данных	Примечание
renter_id	int	Идентификатор арендатора, primary key, not null, auto_increment
fio	varchar(50)	ФИО арендатора, not null

Продолжение таблицы 2.3

Имя поля	Тип данных	Примечание
email	varchar(30)	Адрес эл. почты арендаторы, not null
login	varchar(30)	Логин для авторизации в сервисе, not null
password	varchar(255)	Хэш пароля арендатора для авторизации в сервисе. Метод хэширования описан в таблице «Владельцы», not null
phone	varchar(15)	Телефон пользователя, not null

В таблице 2.4 содержатся данные о бронировании пользователями апартаментов, включая всю историю бронирования.

Пользователь, авторизовавшись, сможет увидеть историю своих броней и управлять текущей бронью (к примеру, прекратить бронь).

Телефон вводится в формате «+7(999)123-25-45» и использоваться во всей системе при отображение в данном виде. В таблице номер хранится без символов «+7».

Таблица 2.4 – Таблица booking (бронирование)

Имя поля	Тип данных	Примечание
booking_id	int	Идентификатор бронирования, primary key, not null, auto_increment
aparts_id	int	Идентификатор апартаментов, которые были забронированы, not null
date_start	date	Дата, начала брони апартаментов, not null
date_end	date	Дата, по которое забронированы апартаменты. Может быть пуста в случае бессрочной брони, null
renter_id	int	Идентификатор пользователя, забронировавшего апартаменты, not null

В таблице 2.5 содержится информация о владельцах апартаментов.

Это список зарегистрированных пользователей, сдающих апартаменты и являющихся их владельцами.

Для шифрования пароля предполагается использовать одностороннее шифрование – лучший способ для сохранения паролей или других важных пользовательских данных. «Одностороннее» значит, что вы можете преобразовать данные в зашифрованную строку, но обратное преобразование невозможно.

В Laravel каждое шифрование одной и той же строки приводит каждый раз к разным результатам. Тем самым для шифрования необходимых данных будет использоваться соответствующая функция с добавлением в начало и конец (перед шифрованием) «крошек». Проверка на правильность введенного пароля будет выполняться также механизмами Laravel (функцией Check).

Злоумышленник, получив доступ к базе, не сможет расшифровать пароль, и придется для взлома выполнять обычный перебор (для чего и доступ к базе не нужен).

Таблица 2.5 – Таблица owners (владельцы)

Имя поля	Тип данных	Примечание
owner_id	int	Идентификатор владельца, primary key, not null, auto_increment
fio	varchar(50)	ФИО владельца апартаментов, not null
email	varchar(30)	Эл. почта владельца апартаментов, not null
login	varchar(30)	Логин для авторизации владельца апартаментов в системе, not null
password	varchar (255)	Хэш пароля владельца апартаментов для авторизации в системе, not null
phone	varchar(15)	Телефон владельца апартаментов, not null

В таблице 2.6 хранится информация о ценах аренды апартаментов за сутки, на сегодня, в том числе и история цен по апартаментам.

Действующей ценой аренды апартаментов является первая цена по этим апартаментам, отсортированная по убыванию даты старта. Тем самым пользователь сможет указать рост цен «на перед», а система автоматически будет брать цену «на сегодня».

Таблица 2.6 – Таблица price (цены)

Имя поля	Тип данных	Примечание
price_id	int	Идентификатор цены, primary key, not null, auto_increment
aparts_id	int	Идентификатор апартаментов, к которым относится данная цена, not null
price	int	Цена за аренду апартаментов в сутки, без учета копеек, поэтому значение будет целочисленным, not null
date_start	date	Дата начала новой цены на апартаменты, not null

В таблице 2.7 содержатся названия файлов изображений апартаментов.

В web-ресурсах изображения хранятся в отдельной папке, доступной извне, чтобы пользователи могли в любой момент открыть отдельно изображение. В данной реализации, доступ к картинкам будет осуществляться по относительному адресу, т. е. в конфигурационном файле, будет прописан путь до папки, хранящей изображения апартамента, а в таблице будет храниться сгенерированное название файла с расширением картинки (*.jpeg, *.jpg, *.png).

Для хранения пути до изображения дома, используется константа IMG_HOUSE = \images\house\, к которой добавляется папка, название которой формируется по шаблону: [ID улицы] + «-» + [номер дома и буква номера дома].

Таблица 2.7 – Таблица aparts_images (изображения апартаментов)

Имя поля	Тип данных	Примечание
aparts_images_id	int	Идентификатор файла названия изображения, primary key, not null, auto_increment
aparts_id	int	Идентификатор апартаментов, к которым относится данный файл изображения, not null
name	varchar(50)	Название файла изображения апартаментов, not null
order_number	int	Порядок сортировки картинок при отображении, null

В таблице 2.8 содержатся апартаменты системы.

Таблица 2.8 – Таблица aparts (апартаменты)

Имя поля	Тип данных	Примечание
aparts_id	int	Идентификатор апартаментов, primary key, not null, auto_increment
house_id	int	Идентификатор дома, в котором находятся апартаменты, not null
owner_id	int	Идентификатор владельца апартаментов, not null
n_room	int	Количество комнат в апартаментах, not null
number	int	Номер апартаментов, not null
letter	char(1)	Буква номера апартамента, null
area	decial(5,1)	Общая площадь апартаментов, not null
kitchen	decimal(4,1)	Площадь кухни в апартаментах. Кухни может не быть, поэтому могут быть пустыми, null

Продолжение таблицы 2.8

Имя поля	Тип данных	Примечание
check_in_time	time	Время, в которое необходимо заехать в апартаменты, not null
check_out_time	time	Время, в которое необходимо выехать из апартаментов, not null
date_cancel	date	Дата, по которое можно отменить бронь на помещении, null
description	longtext	Текст объявления, not null
view_count	int	Количество просмотров, null
couchette_count	int	Количество спальных мест, not null
is_deleted	bit(1)	Признак апартамента «удалено», null
is_active	bit(1)	Признак апартамента «активировано», null

В таблице 2.9 содержатся интервалы апартаментов.

Содержит интервалы дат, в которые запрещено бронирование апартаментов, к примеру, в связи с ремонтом апартаментов или параллельной сдачей в другом агентстве.

Таблица 2.9 – Таблица intervals (интервалы)

Имя поля	Тип данных	Примечание
interval_id	int	Идентификатор интервалов, primary key, not null, auto_increment
aparts_id	int	Идентификатор апартаментов, к которым относится интервал, not null
interval_start	date	Дата, когда апартамент стал неактивным , not null
interval_end	date	Дата окончания неактивности апартамента, not null

2.4 Описание предметной области и структура сайта

Web-сайт для аренды апартаментов предназначен для привлечения потенциальных арендаторов, предоставления информации о его деятельности и обеспечения связи с клиентами.

Сайт предполагает предоставлять возможность выполнять следующие функции:

- осуществление возможности обмена сообщениями в рамках сайта;
- поиск недвижимости, подходящей под требования арендатора;
- поиск и выдача сведений о владельцах квартир;
- круглосуточная работоспособность;
- предоставление подробной информации о каждом предлагаемом объекте.

В web-сайте можно выделить четыре составляющих: «интерфейс гостя», «интерфейс арендодателя», «интерфейс арендатора», «интерфейс модератора». Структура web-приложения представлена на рисунке 2.3.

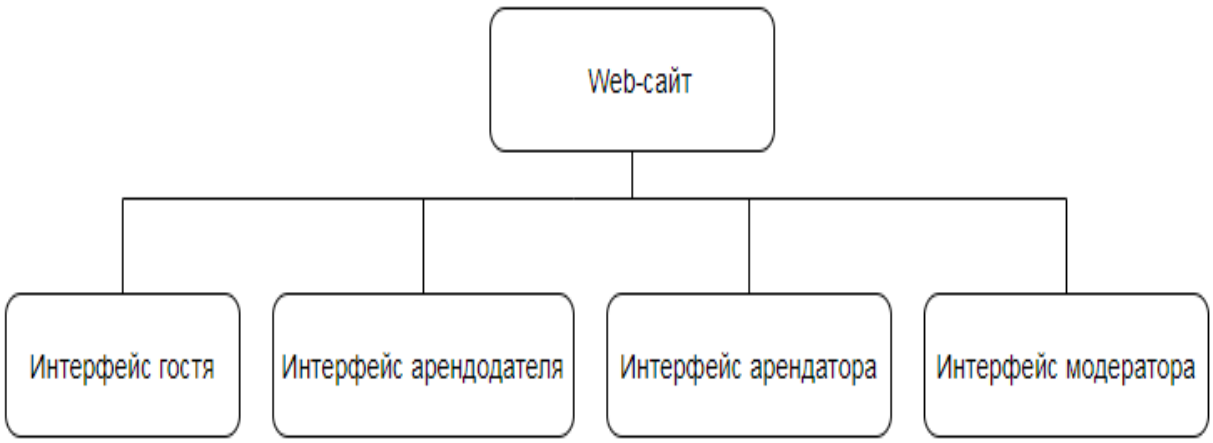

Рисунок 2.3 – Структура web-сайта

«Интерфейс гостя» предназначен для не авторизованных пользователей, им предоставляются возможности только просматривать информацию о компании и информацию об апартаментах.

«Интерфейс арендатора» предоставляет более широкий спектр возможностей. Помимо возможностей предоставленных в «интерфейсе гостя» возможно бронировать объекты, просматривать историю броней,

заполнять контактную информацию в личном кабинете и общаться с арендодателем.

«Интерфейс арендодателя» представляет собой интерфейс необходимый сотруднику, выполняющему роль арендодателя, а именно он имеет право:

- просматривать апартаменты;
- добавлять, удалять и редактировать свои апартаменты в личном кабинете;
- вводить свои данные в личном кабинете;
- обмениваться сообщениями с арендаторами;
- просматривать историю бронирований собственных апартаментов;
- просматривать статистику своих броней.

«Интерфейс модератора» представляет собой интерфейс, в котором модератор имеет право активировать, посмотреть, удалить апартамент, а также написать письмо об ошибке арендодателю, если такая имеется.

Схему страницы сайта целесообразно разделить на три секции (рисунок 2.4):

- верхняя секция – шапка, в которую должны входить:
  1) логотип компании;
  2) кнопки меню;
  3) наименование города;
  4) кнопка авторизации;
  5) кнопка входа для модератора;
- средняя секция, в которой выводится основное содержимое страницы web-сайта;
- нижняя секция – подвал, в котором находятся:
  6) чат;
  7) важная информация;
  8) кнопки социальных сетей.

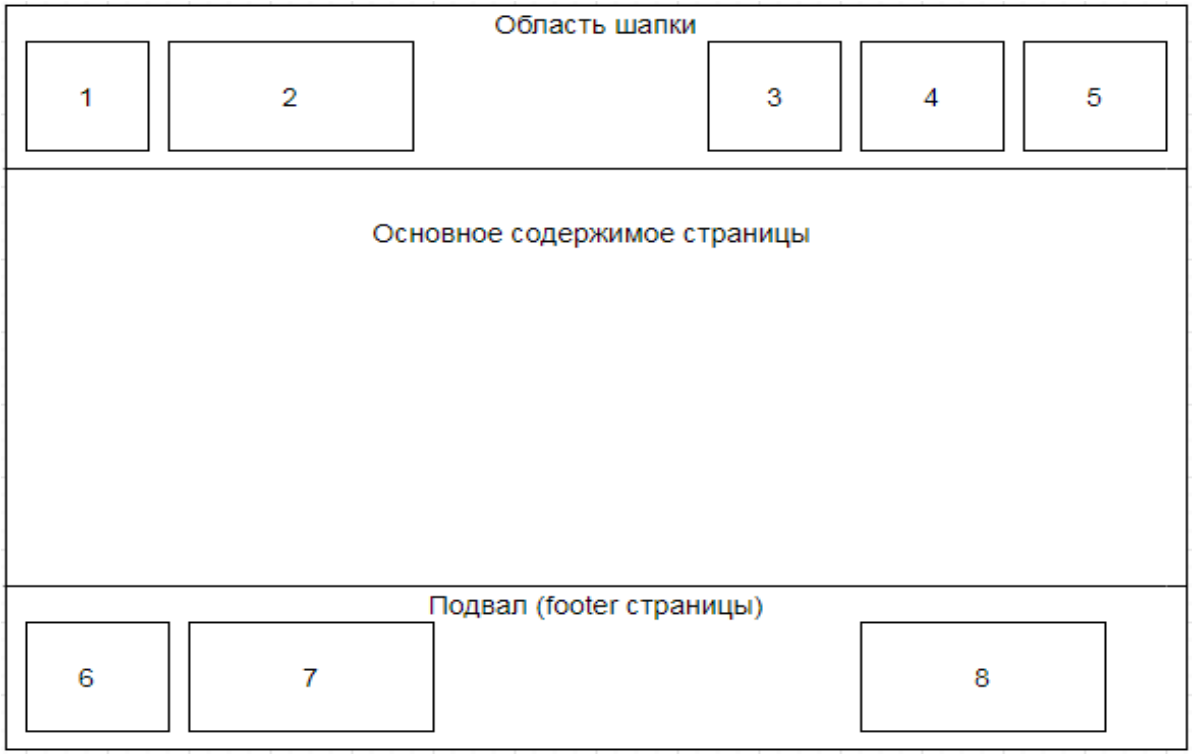

Рисунок 2.4 – Схема страницы

Общедоступные страницы:

1) «Головная» – содержит основную информацию о web-сайте;

2) «Апартаменты» – содержит апартаменты и фильтры для поиска (улица, даты заезда и отъезда, цена и количество спальных мест).

Страницы, предназначенные для арендодателя и арендатора:

1) «Брони» – содержимое страницы разное для разных ролей:

- арендодатель видит свои апартаменты, которые забронировали арендаторы, историю броней его апартаментов (кто и когда бронировал) и статистику, отражающую соответствие максимального количества броней за месяц и количество броней апартаментов арендодателя за месяц;
- арендатор видит активные брони, которые он забронировал, историю своих бронирований, а также статистику своих броней;

2) «Личный кабинет» – содержимое страницы разное для разных ролей:

- арендодатель видит свой профиль, в котором отображены его ФИО, эл. почта, номер телефона, а также его апартаменты, находящиеся в статусе активации (объект не добавится на страницу «Апартаменты», пока его не активирует модератор), которые он может посмотреть или удалить;

• арендатор видит свой профиль, в котором отображены его ФИО, эл. почта и номер телефона;

3) «Сообщения» – содержит чат, в котором арендодатель общается с арендатором.

Страница, предназначенная для модератора:

1) «Управление» – страница для проверки корректности информации, введенной пользователями.

Все страницы не являются статичными кроме страницы «Главная». Определенным пользователям предоставляется возможность вносить в них изменения. Только человек, обладающий определенными правами доступа, имеет право редактировать ту или иную страницу.

Полномочия на доступ предоставляются на основании роли посетителя сайта. На их основании разные пользователи могут посетить одну и ту же страницу, но у одного будет возможность редактировать информацию, а у другого нет.

В таблице 2.10 представлены права доступа к страницам для описанных выше ролей.

Таблица 2.10 – Таблица пользователей и их прав доступа

Роль пользователя	«Головная»	«Апартаменты»	«Брони»	«Личный кабинет»	«Проверка объявлений»	«Сообщения»
Модератор	+	+	–	–	+	–
Арендодатель	+	+	+	+	–	+
Арендатор	+	+	+	+	–	+
Гость	+	+	–	–	–	–

Переход между страницами осуществляется по следующей схеме (рисунок 2.5):

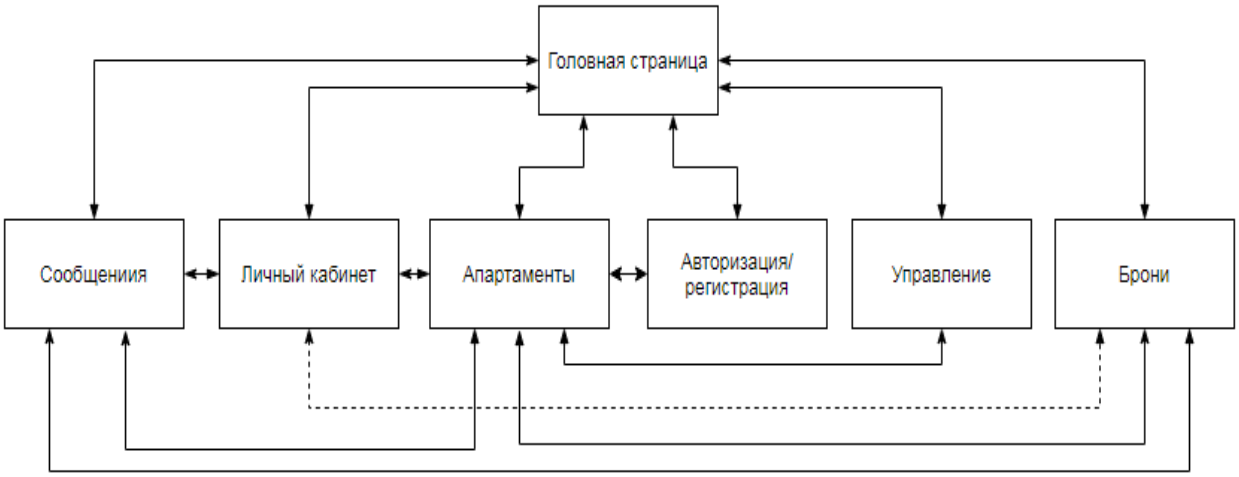

Рисунок 2.5 – Схема переходов страниц

С Головной страницы реализован переход ко всем страницам, существующим на данном web-сайте, а именно можно перейти к страницам «Сообщения», «Личный кабинет», «Апартаменты», «Авторизация/регистрация», «Управление» и «Брони». Со страницы «Сообщения» реализован переход на страницы «Личный кабинет», «Апартаменты», «Брони» и на Головную странницу. Со страницы «Личный кабинет» реализован переход на страницы «Сообщения», «Апартаменты», «Брони» и на Головную страницу. Со страницы «Апартаменты» можно перейти на страницы «Сообщения», «Личный кабинет», «Авторизация/регистрация», «Управление», «Брони» и на Головную страницу. Со страниц «Авторизация/регистрация» и «Управление» осуществлен переход на страницу «Апартаменты» и на Головную страницу. Со страницы «Брони» осуществлен переход на страницы «Сообщения», «Апартаменты», «Личный кабинет» и на Головную страницу

Если неавторизованный пользователь перейдет на страницу «Апартаменты» и попробует арендовать помещение, то ему будет предложено авторизоваться как арендатор, т. е. осуществить переход в «Авторизация/регистрация».

2.5 Выводы

В процессе постановки задачи выделены роли пользователей, которые будут пользоваться web-приложением, страницы сайта. Для каждого пользователя определены права доступа к той или иной странице web-сайта.

На основании ролей и их правах разработан состав и структура сайта, определяющие интерфейс и наполнение всего сайта в целом.

В результате разработки были определены сущности и их атрибуты, на основании которых составлена модель базы данных.

3 РАЗРАБОТКА WEB-САЙТА ДЛЯ АРЕНДЫ АПАРТАМЕНТОВ

3.1 Файл констант и параметров сервиса

В Laravel предусмотрен файл .env для хранения параметров, используемых по всему проекту.

Для получения значения параметра используется функция env(‘ключ параметра’).

В файле находится информация о сайте: название, ссылки на соц. сети, настройка доступа к БД (хостинг, порт, логин, пароль), настройки почтовой службы и прочее. Так же этот файл можно самостоятельно дополнять своими параметрами по мере надобности.

Примерный вид параметров (рисунок 3.1):

```
DB_CONNECTION=mysql
DB_HOST=******.beget.tech
DB_PORT=3306
DB_DATABASE=**********
DB_USERNAME=**********
DB_PASSWORD=*U7**&fI

BROADCAST_DRIVER=log
CACHE_DRIVER=file
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120
```

Рисунок 3.1 – Примерный вид параметров

3.2 Шаблон сайта

Laravel позволяет создать шаблон и позволяет, наследуя его использовать на других страницах. Тем самым при реализаци создан основной шаблон проекта используемый всеми страницами.

Некоторые операторы при формировании html шаблона в Laravel:

- @include() загружает в указанный участок шаблона существующий html документ;

- @extends() наследует содержимое указанного шаблона;
- @yield() служит своеобразным маркером, на место которого будет подставлено содержимое @section() дочернего шаблона;
- @section() @endsection указывает на начало и конец html кода, который будет вставлен в @yield наследуемого шаблона;
- @csrf создает в html документе элемент token, служащий ключом безопасности при POST запросах;
- @empty @else @endempty является проверкой на существование;
- @if … @else @endif оператор условия если;
- {{ }} своеобразная замена <?php .. ?> указывает на участки шаблона, в которых будут выполняться методы Laravel;
- @php @endphp позволяет писать php код внтури шаблона страницы.

Описание схемы корневого шаблона (рисунок 3.2):

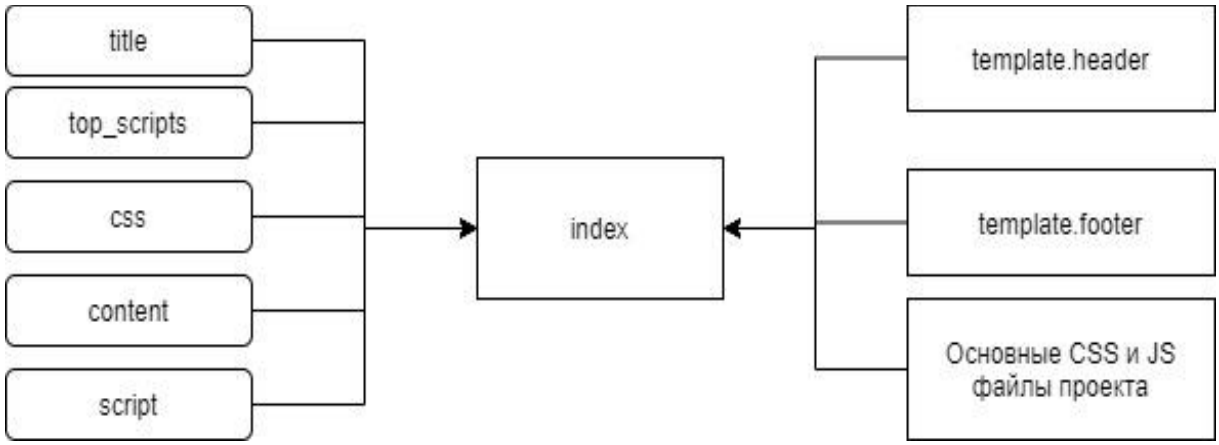

Рисунок 3.2 – Схема корневого шаблона

Основным файлом шаблона является файл index.blade.php. В нем содержатся 5 меток для динамического добавления и статично подключенные файлы проекта.

Для уменьшения количества кода в рамках одного файла шапка и нижняя часть сайта вынесены в файлы template.header.blade.php и template.footer.blade.php. Данные html блоки подключены к основному шаблону и в случае редактирования, к примеру, шапки сайта, изменения применятся у всех страниц, наследуемых index.blade.php.

К основным CSS файлам относятся:

- Font Awesome – иконки сайта;
- Bootstrap – web набор заготовок стилей;
- MDB – модифицированные стили Bootstrap;
- style – дополнительные стили, которые отсутствовали в вышеприведенных файлах или требовали изменения вида отображения.

К основным JS файлам относятся:

1) требуемые файлы для работы стилей bootstrap и mdb:

- jquery-3.4.1;
- popper;
- bootstrap;
- mdb;

2) файлы, написанные в процессе разработки сервиса:

- public;
- modal_form;

3) файл для работы messenger-а Вконтакте:

- https://vk.com/js/api/openapi.js.

Метки шаблона

1. Title – метка для указания названия страницы сайта, отображаемая в браузере (рисунок 3.3):

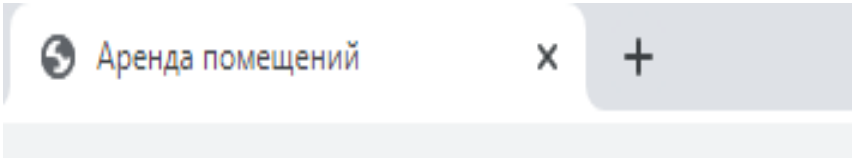

Рисунок 3.3 – Метка Title

2. Top_scripts – метка подключения JS скриптов, без которых дальнейшее формирование отображения html содержимого страницы будет некорректным. К примеру, для заблаговременного отображения формата номера телефона 9997621232 в виде +7 (999) 762-12-32 и т. п.

3. Css – метка для дополнительного подключения стилей, не относящихся к основному проекту и требуемые к подключению в единичных случаях.

4. Content – метка основного содержания сайта. В ней отображается все, что не относится к шапке и нижней части сайта.

5. Script – метка для JS файлов, требуемых для работы сайта и не обязательных для загрузки в самом начале страницы (к примеру, отслеживание событий изменения дат бронирования или нажатия кнопок). В приоритете добавлять скрипты в эту метку, т. к. это ускорит прорисовку страницы и позволит пользователю, не дожидаясь загрузки не важных скриптов, приступить к работе с содержимым страницы (к примеру, к подбору апартаментов).

Тем самым, каждая последующая созданная страница, наследует основной шаблон с заранее подключенными стилями и скриптами.

Текст шаблона проекта (рисунок 3.4, 3.5):

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1,
shrink-to-fit=no">
    <meta http-equiv="x-ua-compatible" content="ie=edge">

    <meta name="description" content=""/>
    <meta name="keywords" content=""/>
    <meta name="robots" content="index,follow"/>

    <meta name="csrf-token" content="{{ csrf_token() }}">
    <title>@yield('title')</title>

    <!-- Scripts -->
    <script src="/js/jquery-3.4.1.min.js"></script>
    <script src="/js/popper.min.js"></script>
    <script src="/js/bootstrap.min.js"></script>

    @yield('top_scripts')
<link href="{{ asset('css/app.css') }}" rel="stylesheet">
    <link          href="/css/fontawesome/css/all.min.css?version=2"
rel="stylesheet" type="text/css">
    <!-- Bootstrap core CSS -->
    <link href="/css/bootstrap.min.css" rel="stylesheet">
    <!-- Material Design Bootstrap -->
    <link href="/css/mdb.min.css" rel="stylesheet">
    <!-- Your custom styles (optional) -->
```

Рисунок 3.4 – Шаблон проекта

```
<link href="/css/style.css" rel="stylesheet">
    <!-- Your custom styles (optional) -->


    <link href="/css/template_super_style.css" rel="stylesheet">
    @yield('css')


</head>
<body>

@include('template.header')

<div class="wrapper d-flex align-items-stretch mt-3">


    <div class="container-fluid">

        @yield('content')
    </div>

</div>
<div class="" style="height: 50px"></div>

@include('template.footer')
<div id="mf"></div>{{-- для вставки модального окна --}}

<script src="/js/mdb.min.js"></script>
<script src="/js/modal_form.js"></script>
<script src="/js/public.js" charset="utf-8"></script>
@yield('script')

<script> {{-- // Animations initialization --}}
    new WOW().init();
</script>
<script type="text/javascript"
src="https://vk.com/js/api/openapi.js?168"></script>
<!-- VK Widget -->
<div id="vk_community_messages"></div>
<script type="text/javascript">
    VK.Widgets.CommunityMessages("vk_community_messages", *********, {
        widgetPosition: "left",
        tooltipButtonText: "Мы можем помочь Вам с выбором"
    });
</script>

</body>
</html>
```

Рисунок 3.5 – Продолжение рисунка 3.4

3.3 Разработка инструментальных средств

3.3.1 Модальные окна

При авторизации, подтверждении действий (к примеру, удаление апартаментов), указании города, при создании апартаментов было использовано динамически формируемое модальное окно. Графически оно разделено на 3 части и может содержать в теле окна любую информацию: от информационного сообщения, до формы с вводом данных (рисунок 3.6):

Рисунок 3.6 – Модальное окно

Blade-файл шаблона ($url – адрес, где находится выполняемый контроллер; $method – способ отправки запроса POST или GET; $name – название окна; $body – тело модального окна; $buttons – дополнительные кнопки-действия) (рисунок 3.7):

```
<div class="modal fade right" id="modal_form" tabindex="-1" role="dialog"
aria-labelledby="modal_form"
     aria-hidden="true">
    <form action="{{    @$url    }}" method=@empty(!@$method) {{"$method"}}
@else "post" @endempty>
        @csrf
        <div class="modal-dialog modal-lg" role="document">
            <div class="modal-content">
                <div class="modal-header">
                    <h5 class="modal-title" id="modal_form_name">{!! $name
!!}</h5>
                    <button type="button" class="close" data-dismiss="modal"
aria-label="Close">
                        <span aria-hidden="true">&times;</span>
                    </button>
                </div>
                <div class="modal-body">
                    {!! $body !!}
                </div>
                <div class="modal-footer col-12">
                        {!! $buttons !!}
                        <button type="button" class="btn btn-secondary align-
right" data-dismiss="modal">Закрыть</button>
                </div>
            </div>
        </div>
    </form>
</div>
```

Рисунок 3.7 – Blade-файл шаблона

На основном шаблоне проекта расположен div блок с id=«mf». Именно в него, с помощью JS функции встраивается готовое окно для открытия модального диалога.

Замена блока (рисунок 3.8) с id=«mf» описана в файле modal_form.js:

```
function mf_open($method, $url, $data_form) {
    $.ajax({
        async: false,
        headers: {'X-CSRF-TOKEN': $('meta[name="csrf-
token"]').attr('content')},
        type: $method,
        url: $url,
        data: {data_form: $data_form},
        success: function (data) {
            if (data.show_modal == true){
                document.getElementById("mf").innerHTML = data.mf_form;
                $('#modal_form').modal('show');
            }
        },
        error: function () {
            console.log('error from modal form');
        },
    });
}
```

Рисунок 3.8 – Замена блока

3.3.2 Отправка запросов без перезагрузки страницы (AJAX)

Имеет место быть отправка запросов, без перезагрузки страницы. Для этого была написана функция, которая отправляет запрос по указанному URL, передавая в него параметры data_form (рисунок 3.9). Данными параметрами могут быть как значения из формы, так и любые параметры переданные в функцию, тем самым остается только написать обработчик входящего запроса.

```
function sendQuery($method, $url, $data_form){
    let r = undefined;
    $.ajax({
        async: false,
        headers: {'X-CSRF-TOKEN': $('meta[name="csrf-
token"]').attr('content')},
        type: $method,
        url: $url,
        data: {data_form: $data_form},
        success: function (data) {
            r = data;
        },
        error: function () {
            console.log('error from sendQuery');
        },
    });
    return r;
}
```

Рисунок 3.9 – Функция, отправляющая запрос по указанному URL

При реализации параметрами являлись сериализованные значения на форме, на которой находился input (рисунок 3.10). К примеру, при указании периодов брони, система отправляет запрос для проверки – свободны ли апартаменты на указанный период дат:

```
$( "input[type='date']" )
    .change(function () {

        let data_form = $(this.form).serializeArray();
        let req = sendQuery('POST', '{{route('booking.check_date')}}',
data_form);
        document.getElementById("date_check").innerHTML = req.res;
        document.getElementById("SendItem").disabled = req.res !== '';
    })
    .change();
```

Рисунок 3.10 – Запрос на проверку дат брони и вывод ответа

3.3.3 Отправка сообщений на электронную почту

Для отправки сообщений на почту пользователю создана функция отправки сообщения MyMail (рисунок 3.11).

```
<?php

namespace App\Http\Helpers;

use Illuminate\Support\Facades\Mail;

class MyMail
{
  public static function sendMail($recipients, $mail_name, $mail_body,
$sender = null): bool
  {
    if (!empty($recipients)) {
      Mail::send('template.emails.default', ['body'=>$mail_body], function
($message) use ($mail_name, $recipients, $sender) {
        if (empty($sender)) {
          $sender = env('MAIL_USERNAME');
        }
        $message->from($sender, env('MAIL_USERNAME'));
        $message->to($recipients, 'Receiver')->subject($mail_name);
      });
    }
    return true;
  }
}
```

Рисунок 3.11 – Функция отправки сообщения

В данном классе текст письма вставляется в заранее созданный шаблон (template.emails.default), содержащий шапку, контактную информацию сервиса, с которого было отправлено сообщение.

Для отправки сообщения от имени сервиса необходимо указать получателя, тему письма, содержимое письма. Указав email отправителя в $sender, сообщение будет отправлено от имени отправляющего, иначе отправитель будет тот, кто указан в параметрах проекта (в *.env файле фреймворка, в нем Laravel ищет базовые параметры).

Для отправки сообщений был настроен почтовый сервис, параметры которого указаны в *.env файле фреймворка (рисунок 3.12).

```
MAIL_DRIVER=smtp
MAIL_HOST=smtp.beget.com
MAIL_PORT=****
MAIL_USERNAME=**********.ru
MAIL_PASSWORD=******kD*w
MAIL_ENCRYPTION=null
MAIL_ADMIN_EMAIL=admin@******.ru
```

Рисунок 3.12 – Параметры почтового сервиса

3.4 Общедоступные страницы

3.4.1 Головная страница

На рисунке 3.13 изображена головная страница web-сайта, на которую имеют доступ все посетители сайта. На ней содержатся данные о том, какую информацию можно получить на сайте компании. С этой страницы можно перейти на все страницы, реализованные на сайте.

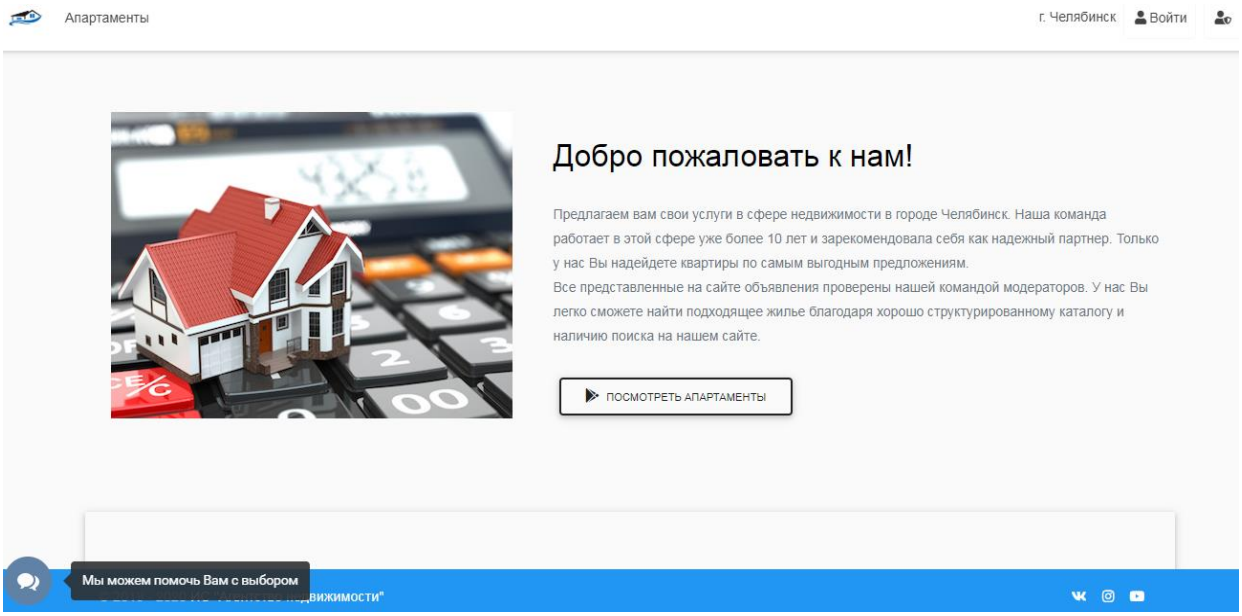

Рисунок 3.13 – Головная страница

3.4.2 Страница «Апартаменты»

Страница «Апартаменты» (рисунок 3.14) содержит подробную информацию о предоставляемых объектах; здесь происходит процесс поиска и бронирования апартаментов:

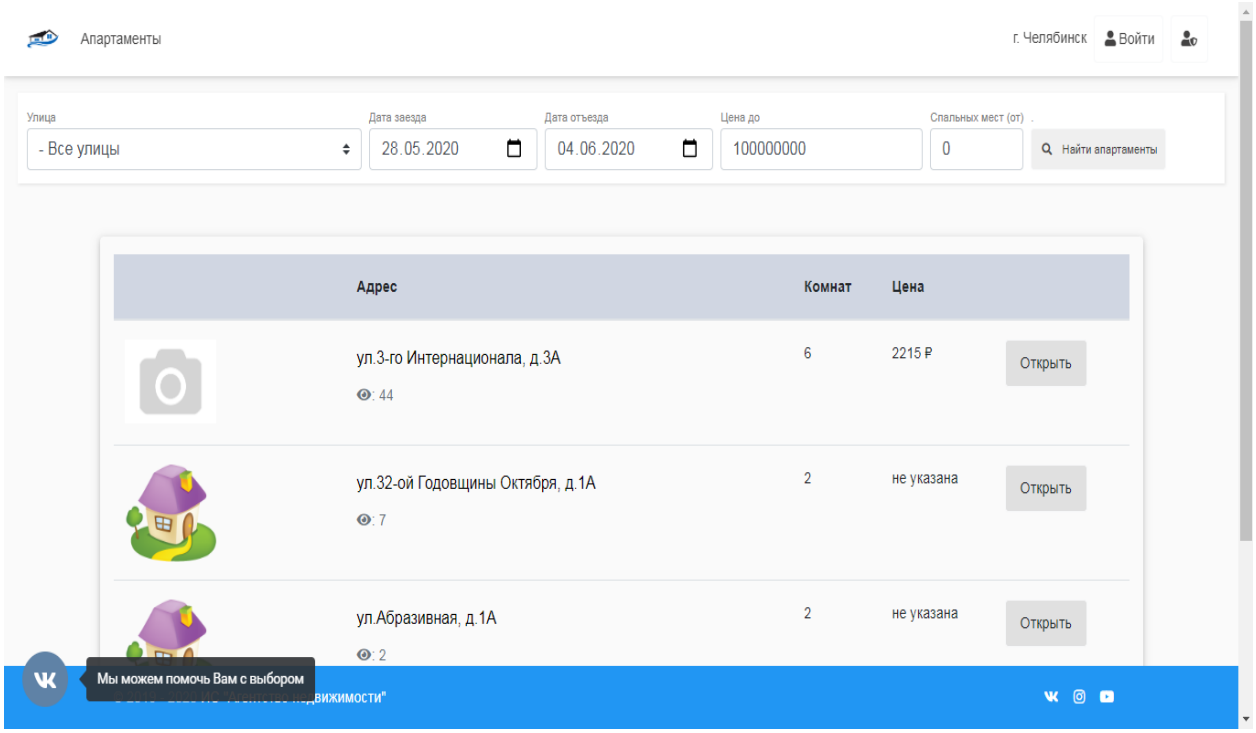

Рисунок 3.14 – Страница Апартаменты

3.5 Страницы, предназначенные для арендодателя и арендатора

3.5.1 Страница «Брони»

«Брони» – содержимое страницы разное для разных ролей.

1. Арендодатель видит свои апартаменты, которые забронировали арендаторы (рисунок 3.15), историю броней его апартаментов (кто и когда бронировал) (рисунок 3.16) и статистику, отражающую соответствие максимального количества броней за месяц и количество броней апартаментов арендодателя за месяц (рисунок 3.17).

2. Арендатор видит активные брони, которые он забронировал, историю своих бронирований, а также статистику своих броней.

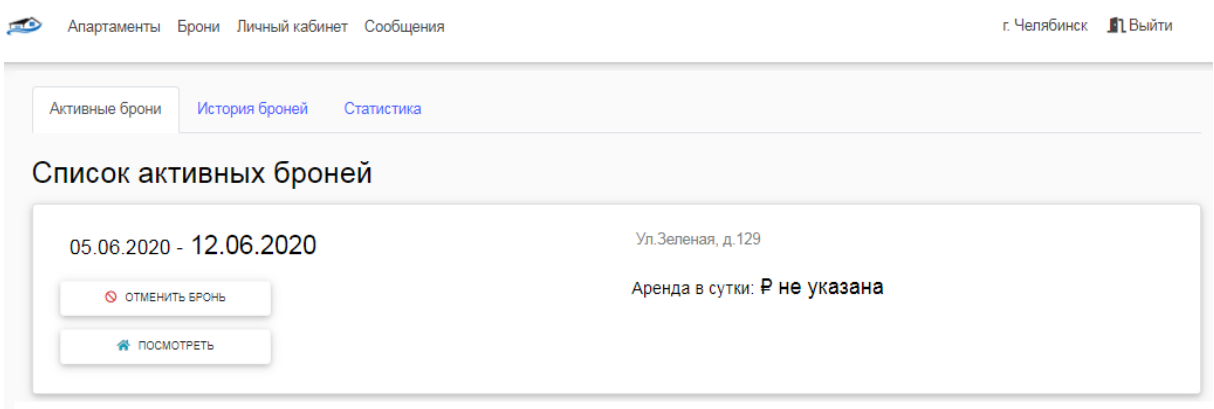

Рисунок 3.15 – Активные брони

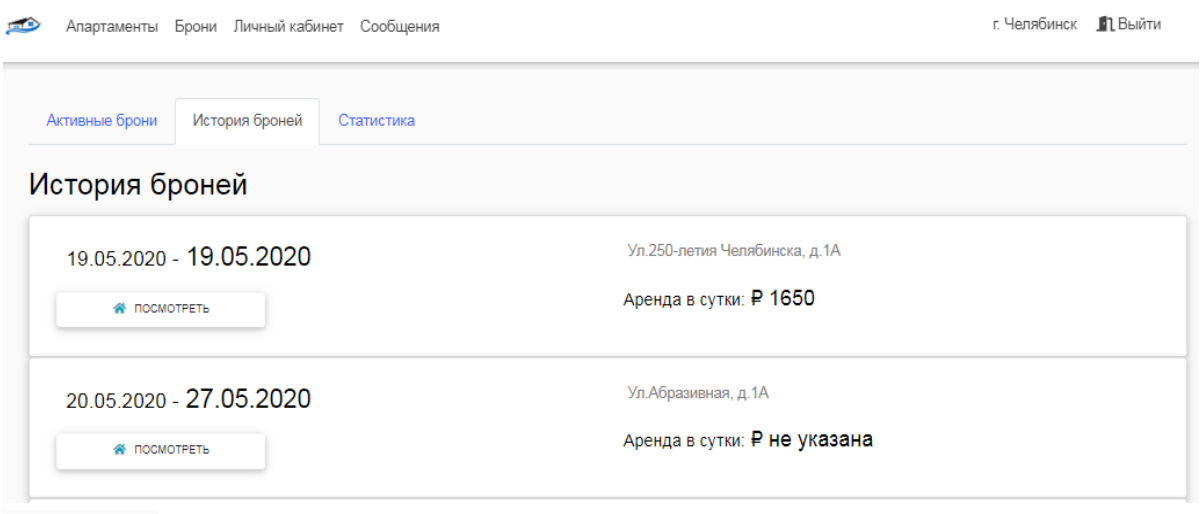

Рисунок 3.16 – История броней

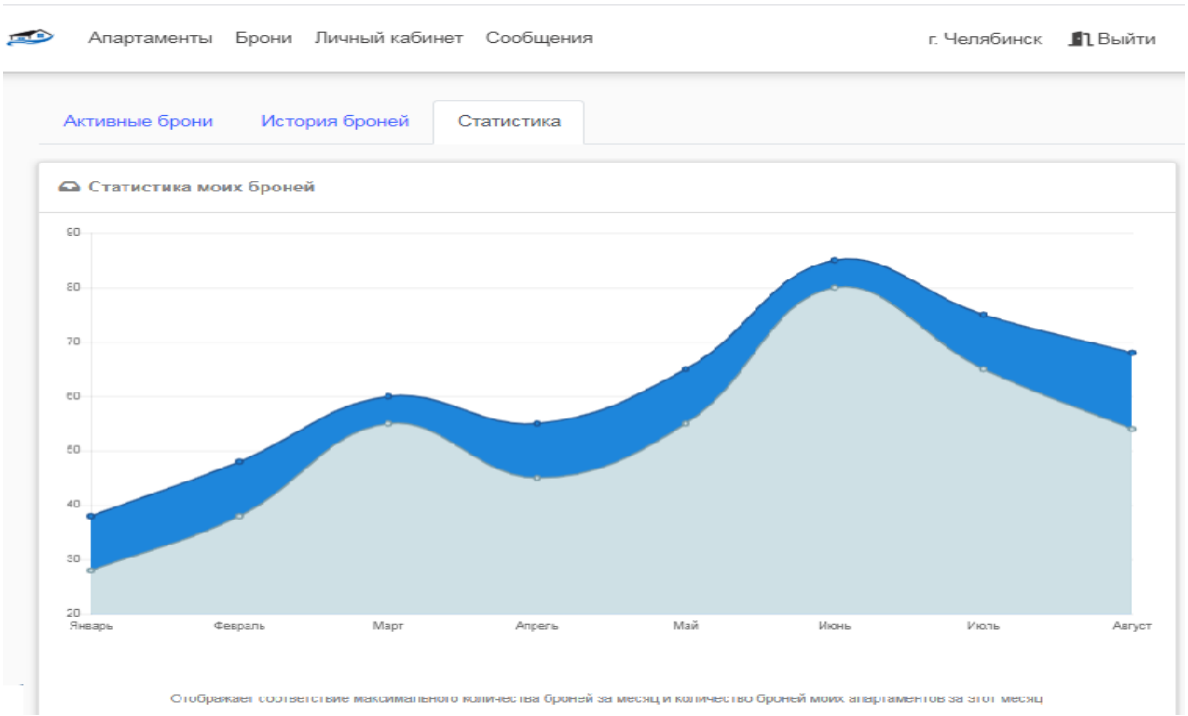

Рисунок 3.17 – Статистика

3.5.2 Страница «Личный кабинет»

«Личный кабинет» – содержимое страницы разное для разных ролей

1. Арендодатель видит свой профиль (рисунок 3.18), в котором отображены его ФИО, эл. почта, номер телефона, а также его апартаменты (рисунок 3.19), находящиеся или не находящиеся в статусе активации (объект не добавится на страницу «Апартаменты», пока его не активирует модератор), которые он может посмотреть или удалить.

2. Арендатор видит свой профиль, в котором отображены его ФИО, эл. почта и номер телефона.

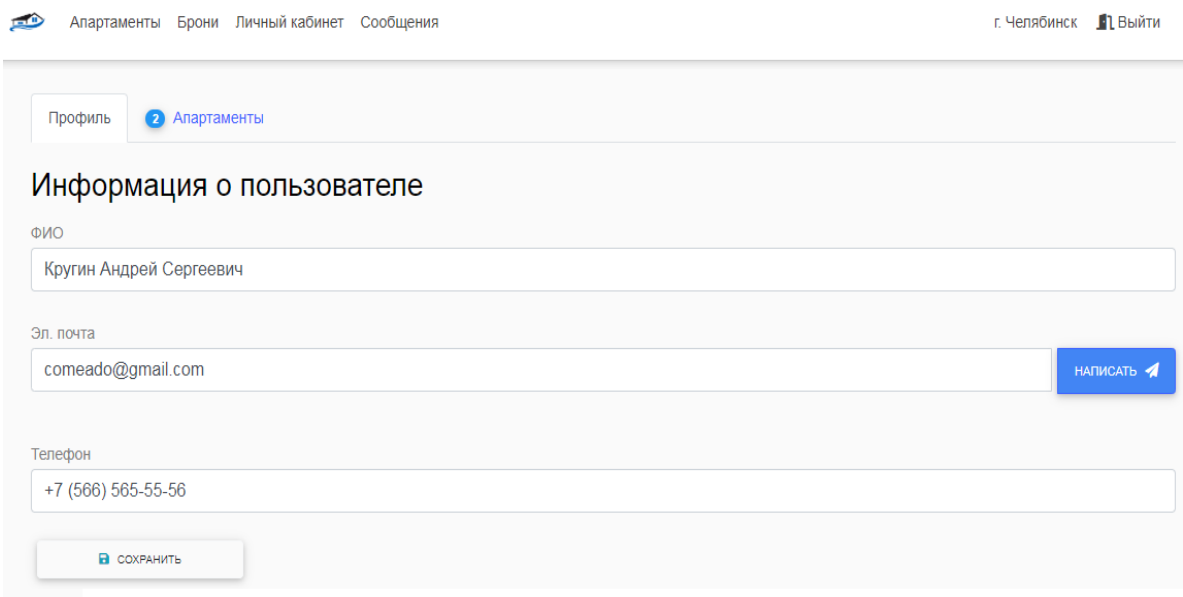

Рисунок 3.18 – Профиль

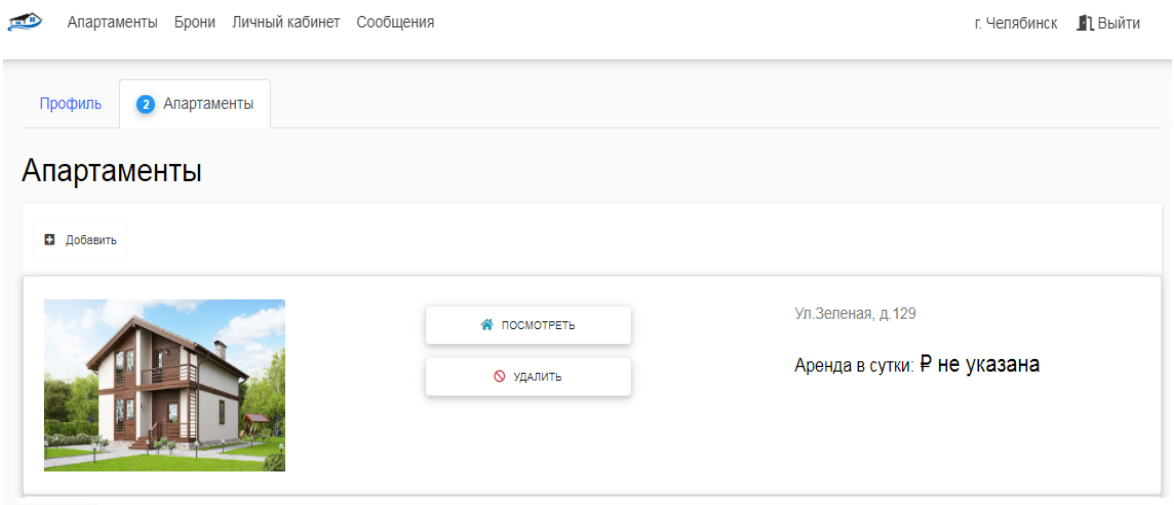

Рисунок 3.19 – Апартаменты арендодателя

3.5.3 Страница «Сообщения»

«Сообщения» – содержит диалоговое окно для общения арендодателя с арендатором (рисунок 3.20).

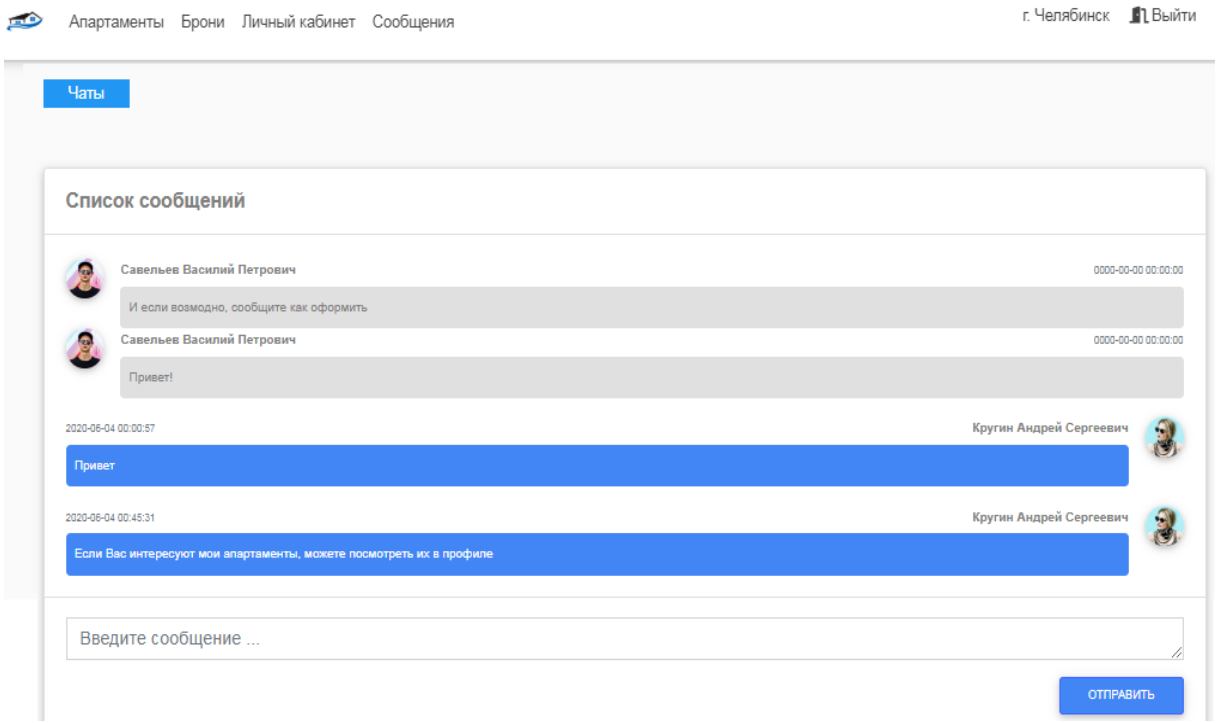

Рисунок 3.20 – Окно для общения арендодателя с арендатором

3.6 Страница, предназначенная для модератора

«Управление» – страница проверки объявления, в которой модератор будет нажимать «Проверено» и объявление будет отображаться на сайте только после того, как он это сделает (страница 3.21).

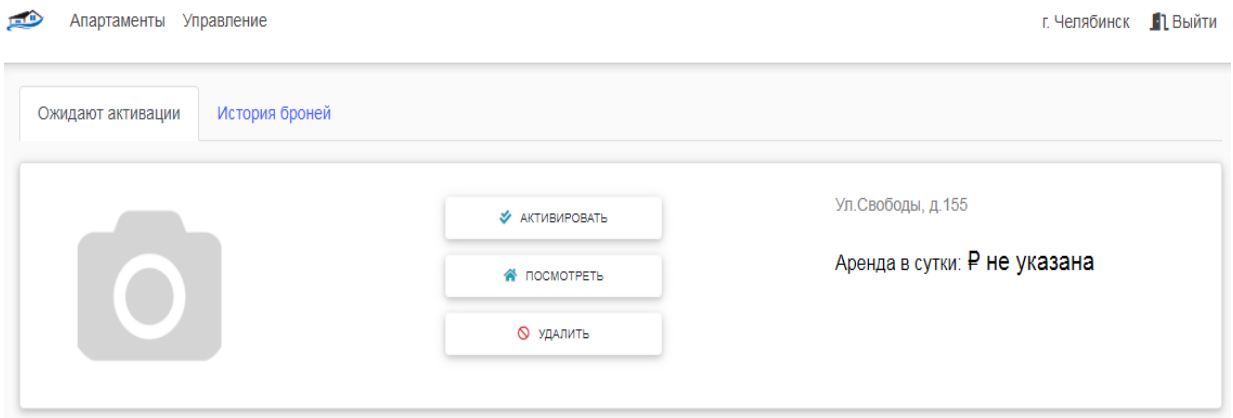

Рисунок 3.21 – Страница управление

3.7 Регистрация и авторизация

3.7.1 Регистрация

Для регистрации арендодателю/арендатору необходимо нажать кнопку Войти → Зарегистрироваться.

Откроется страница с формой регистрации (рисунок 3.22). Пользователь заполняет поля, и нажав на кнопку «Зарегистрироваться», система проверяет все ли обязательные поля заполнены, и если необходимо, подсказывает, какое поле необходимо заполнить.

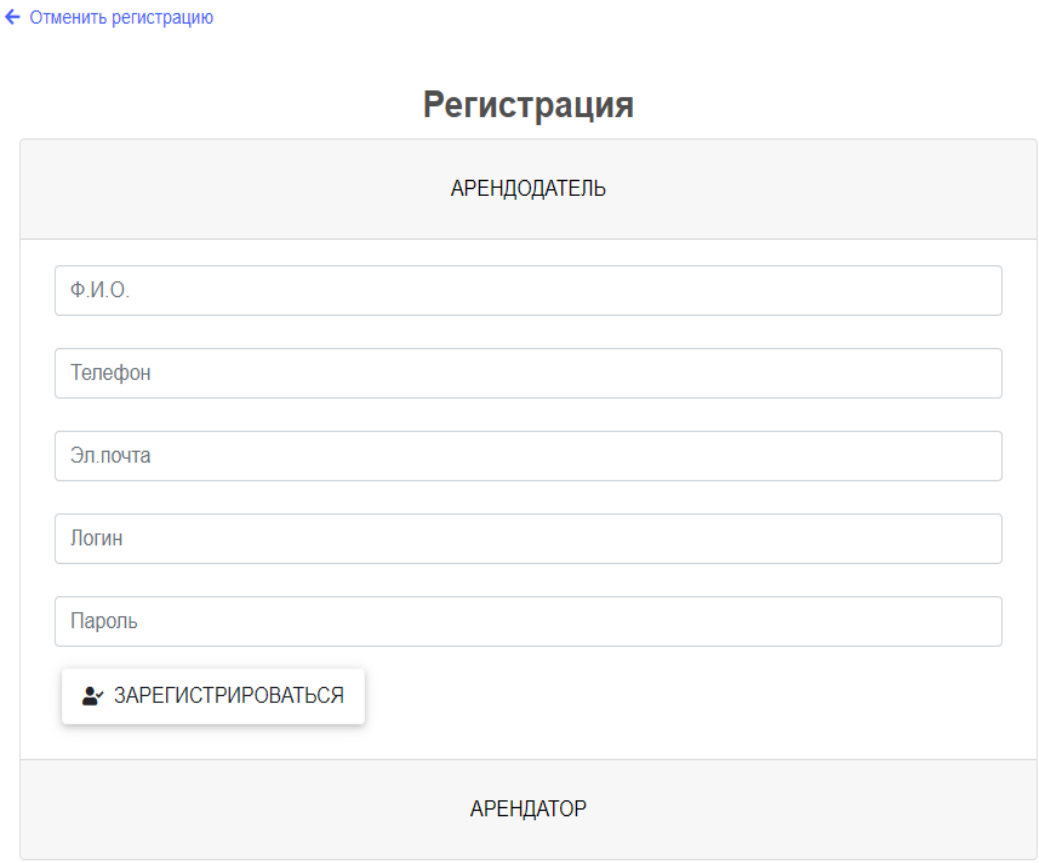

Рисунок 3.22 – Форма регистрации арендодателя

Нажав кнопку «Зарегистрироваться» и успешно пройдя проверку заполненности полей, сервис выполняет регистрацию пользователя, отправляет на указанную эл. почту сообщение с логином (рисунок 3.23).

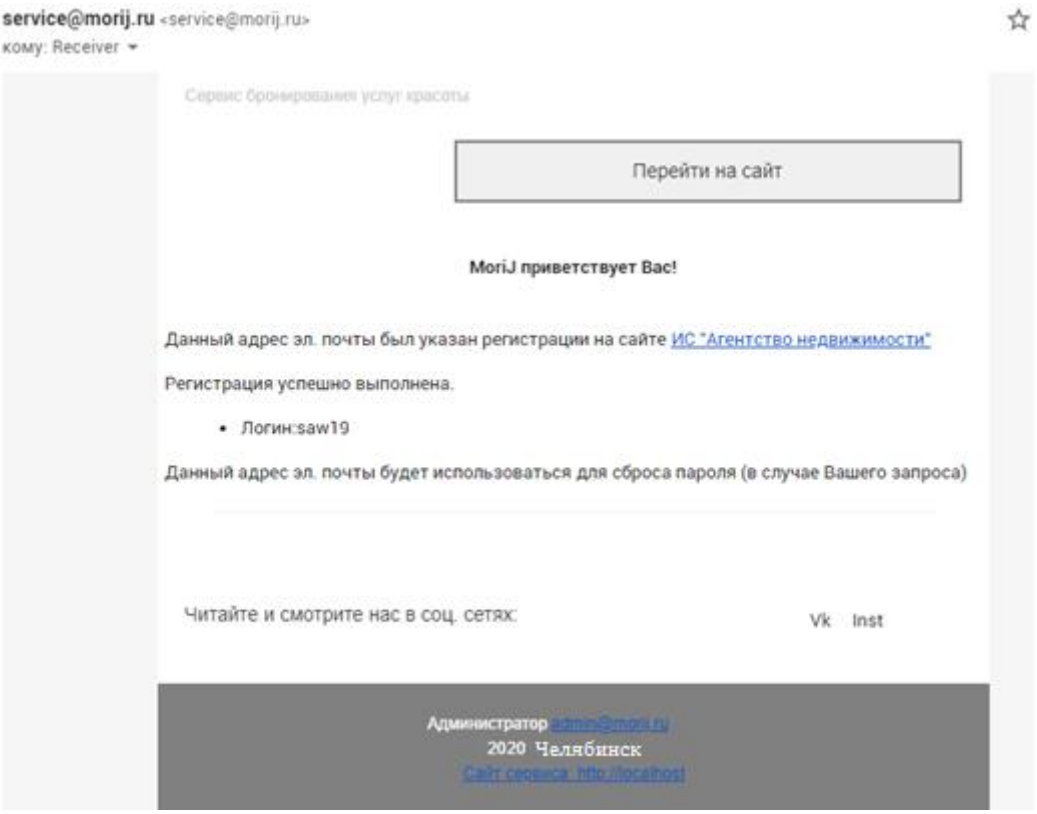

Рисунок 3.23 – Письмо от сервиса

3.7.2 Авторизация

Авторизация разделена на 2 категории: вход для модератора, вход для арендодателей/арендаторов.

Для удобства кнопка авторизации модератора вынесена на главную страницу и имеет иконку (рисунок 3.24):

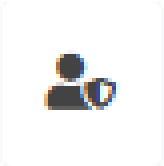

Рисунок 3.24 – Иконка модератора

У модератора имеется единый логин и пароль.

Для авторизации пользователя используется модальное окно, открываемое по нажатию кнопки (рисунок 3.25):

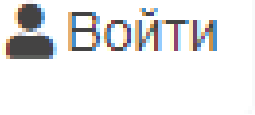

Рисунок 3.25 – Иконка для входа в модальное окно авторизации

При авторизации необходимо ввести в открывшемся окне логин, пароль и указать, под какой ролью необходимо авторизоваться (Арендодатель/Арендатор) (рисунок 3.26):

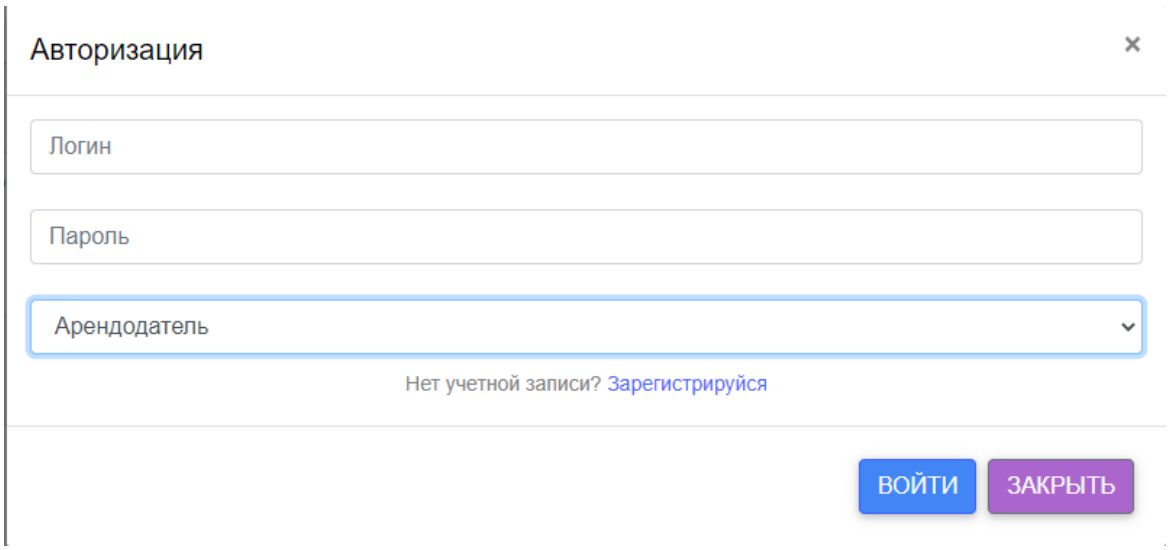

Рисунок 3.26 – Модальное окно авторизации

В случае неверной авторизации, система сообщит о неверном логине/пароле, выводя в правом нижнем углу окно с информационным сообщением (рисунок 3.27):

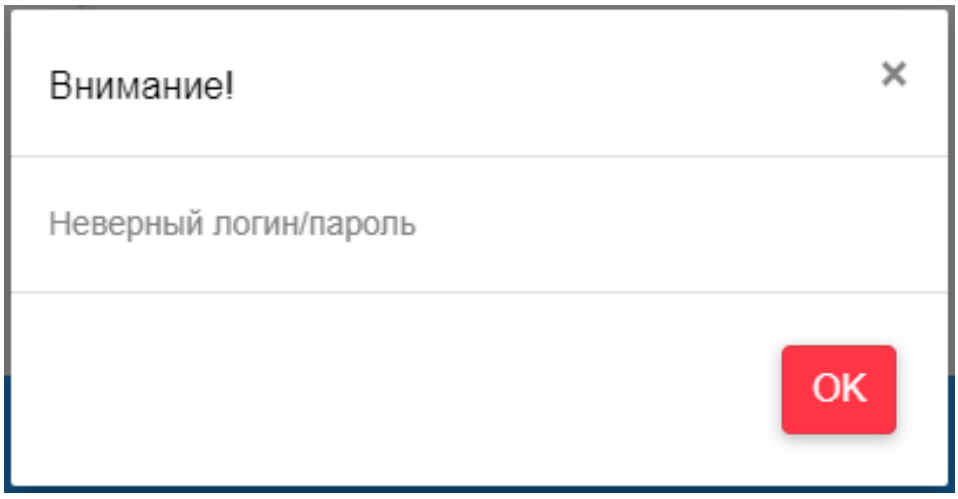

Рисунок 3.27 – Сообщение об ошибке

При успешной авторизации данные о пользователе записываются в сессию и используются при работе проекта.

Метод авторизации описан в контроллере AuthController.php и имеет вид (рисунок 3.28):

```
public function mf_auth_set(Request $request)
{
    $ip = request()->ip();
    $type_user=  Request('type_user'); /*owner  or renter*/
    $login =    Request('login');
    $pass =     Request('pass');
    $auth =     App\ExecProcedure::execProc('p_auth_get', [$type_user,
$login, $ip]);

    if (empty($auth)){// Если не найдена учетная запись
        $msg = 'Данный логин не существует.<br>Верно ли указано поле:
Арендатор/Арендодатель?';
        setcookie( 'msg_error',  $msg );
    }
    else{
        $auth = collect($auth[0])->toArray();
        $isAuth =  App\GlobalProcedure::checkHash($pass,$auth['password']) ;
        if (!$isAuth) {
            $msg = 'Неверный логин/пароль';
            setcookie( 'msg_error',  $msg );
        }
        else{
            // Успешно авторизовались
            Session::put('users.id', $auth['users_id']);
            Session::put('users.email', $auth['email']);
            Session::put('users.phone', $auth['phone']);
            Session::put('users.login', $auth['login']);
            Session::put('users.type', $auth['user_type']);
            Session::put('users.FIO', $auth['FIO']);
        }
    }
    App\GlobalAction::setRights();
    return redirect(route('home'));
}
```

Рисунок 3.28 – Метод авторизации

3.8 Выводы

В ходе работы описаны методы и средства реализации сайта. В процессе разработки сайта разработаны его архитектура, классы и функции. Проведена отладка и тестирование, создан сайт. Приобретены навыки разработки веб-сервисов на языке программирования PHP в фреймворке Laravel.

ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы разработано программное обеспечение для аренды и бронирования апартаментов, реализованное в форме web-сайта.

В процессе работы над проектом решены следующие задачи:

- изучение предметной области;
- формулировка требований к программному обеспечению и бизнес-процессов при выполнении аренды;
- определение ролей и функций участников бизнес-процессов;
- проектирование базы данных;
- разработка структуры сайта;
- разработка функционала пользователей сайта;
- написание, отладка и тестирование программного кода, реализующего необходимые функции;
- публикация сайта в интернете.

Таким образом, цель проекта достигнута, а именно: разработана программа «аренда апартаментов».

В дальнейшем предполагается расширение возможностей данного приложения, например:

- осуществление поиска с помощью карты;
- увеличение фильтров для поиска;
- предоставление возможности оставлять свои отзывы и пожелания.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Авито [Электронный ресурс]. – Режим доступа: https://www.avito.ru/ (дата обращения 15.01.2020).

2 Атре, Ш. Структурный подход к организации баз данных / Ш. Атре. – М.: Финансы и статистика, 1983. – 317 с.

3 Боуман, Дж.С. Практическое руководство по SQL / Дж.С. Боуман, С.Л. Эмерсон. – М.: Диалектика, 2005. – 352 с.

4 Букинг.ком [Электронный ресурс]. – Режим доступа: https://www.booking.com/ (дата обращения 26.01.2020).

5 ВикипедиЯ [Электронный ресурс]. – Режим доступа: http://ru.wikipedia.org/wiki / (дата обращения 12.04.2020).

6 Гаевский, А.Ю. 100% самоучитель. Создание Web-страниц и Web-сайтов. HTML и JavaScript / А.Ю. Гаевский, В.А. Романовский. – М.: Триумф, 2015. – 464 с.

7 Грабер, М. Введение в SQL: Пер. с англ. / М. Грабер. – М.: Лори, 1996. – 380 с.

8 Дронов, В.А. PHP, MySQL, HTML5 и CSS 3 Разработка современных динамических Web-сайтов / В.А. Дронов. – Спб.: BHV, 2016 – 688 с.

9 Едомский, Ю.Е. Техника Web-дизайна для студента / Ю.Е. Едомский. BHV–СПб, 2008. – 392 с.

10 Карпова, Т.С. Базы данных: модели, разработка, реализация / Т.С. Карпова. – СПб.: Питер, 2001. – 304 с.

11 Колисниченко, Д. Н. Профессиональное программирование на PHP: Монография / Д.Н. Колисниченко. – М.: ООО «Бином», 2007. – 512 с.

12 Компаньон [Электронный ресурс]. – Режим доступа: http://www.realtory.ru/ (дата обращения 22.02.2020).

13 Конверс, Т. PHP 5 и MySQL. Библия пользователя / Т. Конверс, Д. Парк, К. Морган // М.: Вильямс, 2009. – 1216 с.

14 Котеров, Д.В. PHP / Д.В. Котеров, А.А. Костарёв. – Спб. BHV, 2014. – 1104 с.

15 Кренке, Д. Теория и практика построения баз данных: [пер.с англ] / Д. Кренке. – 9-е изд. – СПб.: Питер, 2005. – 858 с.

16 Кузнецов, М.В. PHP на примерах / М.В. Кузнецов, И.В. Симдянов. – Спб.: BHV, 2012. – 400 с.

17 Лещев, Д. Создание интерактивного web-сайта: учебный курс / Д. Лещев. – СПб.: Питер, 2011. – 544 с.

18 Маклафлин, Б. PHP и MySQL. Исчерпывающее руководство / Б. Маклафлин. – Спб.: Питер, 2014. – 544 с.

19 Прохоренок, Н.А. HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера / Н.А. Прохоренок // СПб.: БХВ-Петербург, 2010. – 890 с.

20 Стружкин, Н.П. Базы данных: проектирование: Учебник для академического бакалавриата / Н.П. Стружкин, В.В. Годин. – Люберцы: Юрайт, 2016. – 477 с.

21 Форум PHP программистов [Электронный ресурс]. – Режим доступа http://php.ru, свободный.

22 Фуфаев, Э.В. Базы данных / Э.В. Фуфаев, Д.Э. Фуфаев. – М.: Изд. Центр «Академия», 2005. – 320 с.

23 Хабр [Электронный ресурс]. – Режим доступа: https://habr.com /ru/post/267721/ (дата обращения 15.03.2020).

24 Циан [Электронный ресурс]. – Режим доступа: https://cian.ru/ (дата обращения 19.01.2020).

25 Шкрыль, А. PHP – это просто. Программируем для Web-сайта / А. Шкрыль. – М.: БХВ-Петербург, 2015. – 368 с.

26 Шпак, Ю.А. Проектирование баз данных. Просто как дважды два / Ю.А. Шпак. – М.: Эксмо, 2007. – 304 с.

27 Laravel [Электронный ресурс]. – Режим доступа: https://laravel.com/ (дата обращения 14.02.2020).

28 jQuery. Официальный сайт [Электронный ресурс]. – Режим доступа: http://jquery.com, свободный.

29 PHP. Официальный сайт [Электронный ресурс]. – Режим доступа: http://php.net, свободный.

30 Ruseller.com [Электронный ресурс]. – Режим доступа: https:// ruseller.com/lessons.php?id=666 (дата обращения 15.04.2020).

ПРИЛОЖЕНИЕ 1
Контроллеры

1) AuthController

```
<?php

namespace App\Http\Controllers\Auth;

use Illuminate\Support\Facades\Session;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use App;
use App\Http\Helpers\MyMail;

class AuthController extends Controller
{
    public function mf_auth(Request $request) /* открыть окно
авторизации */
    {
        $id_user = 0;
        if (!empty(session('users.id'))) {
            $id_user = session('users.id');
        }
        $uri = $_SERVER['HTTP_REFERER'];
        $uri_orig = $_SERVER['HTTP_ORIGIN'].'/';
        $a = strlen($uri_orig);
        $uri = substr($uri, $a, strlen($uri));

        $route_name =$uri;

        $body = '';
        $name = 'Авторизация';

        $url= route('auth.mf_set');
        $method ='post';

        $buttons_auth = view('t_widget.components.button')-
>with([
            'url' =>  $url,
            'name' => 'Войти',
            'type' => 'submit'
        ]);
        $buttons_auth = $buttons_auth->render();

        $buttons = $buttons_auth ;

        $auth_form =  view('template.base_form.auth')->with([
            'name' => 'vvv',
            'type' => 'submit'
        ]);
        $body .=  $auth_form;

        $html = view('t_widget.modal_form', compact('url',
'name', 'body', 'buttons','method'));
        $html = $html->render();
        return response()->json(['mf_form' => $html, 'show_modal'
```

```
=> true]);
    }

    public function mf_auth_set(Request $request)
    {
        $ip = request()->ip();
        $type_user=  Request('type_user'); /*owner  or renter*/
        $login =    Request('login');
        $pass =     Request('pass');
        $auth =     App\ExecProcedure::execProc('p_auth_get',
[$type_user, $login, $ip]);

        if (empty($auth)){ // Если не найдена учетная запись
            $msg = 'Данный логин не существует.<br>Верно ли
указано поле: Арендатор/Арендодатель?';
            setcookie( 'msg_error',  $msg );
        }
        else{
            $auth = collect($auth[0])->toArray();
            $isAuth =
App\GlobalProcedure::checkHash($pass,$auth['password']) ;
            if (!$isAuth) {
                $msg = 'Неверный логин/пароль';
                setcookie( 'msg_error',  $msg );
            }
            else{
                // Успешно авторизовались, грузим инфу о
пользователе
                Session::put('users.id', $auth['users_id']);
                Session::put('users.email', $auth['email']);
                Session::put('users.phone', $auth['phone']);
                Session::put('users.login', $auth['login']);
                Session::put('users.type', $auth['user_type']);
                Session::put('users.FIO', $auth['FIO']);
            }
        }
        App\GlobalAction::setRights();
        return redirect(route('home'));
    }


    public function registrationSend(Request $request)
/*Регистрация отправка на сервер*/
    {
        $URL_ToACTIVE ='';
        $ip = request()->ip();
        $type_user=  Request('type_user');
        $fio =  Request('fio');
        $phone =  Request('phone');
        $email =  Request('email');
        $login =  Request('login');
        $pass  =  Request('pass');
        $token = App\GlobalProcedure::getRandomToken();
        $pass  = App\GlobalProcedure::getHash($pass);

        $p_reg  = App\ExecProcedure::execProc('p_auth_set',
[$type_user,$login, $pass, $email, $phone, $fio]);
```

```
        $result = False;
        foreach ($p_reg as $row) {
            if ($row->result  == 1){
                $result = True;
                $body_mail = '<p>Данный адрес эл. почты был
указан регистрации на сайте <a href="' . env('APP_URL') .
'">'.env('APP_NAME').'</a></p>';
                $body_mail .= '<p>Регистрация успешно
выполнена.</p>';
                $body_mail .= '<li>Логин:'.$login.'</li>';
                $body_mail .= '<p>Данный адрес эл. почты будет
использоваться для сброса пароля (в случае Вашего запроса)</p>';

                $is_send_mail = MyMail::sendMail($email,
'Регистрация', $body_mail);
            }
            // Todo ОШИБКА - вывести!!!!
            $msg = $row->msg;
        }

        if ($result) {
            setcookie( "msg_error", $msg);
            return view('home.home_index');
        }
        else{
            setcookie( "msg_error", $msg);
            return view('template.base_form.registration',
compact('msg'));
        }
    }

    public function logout_send() /*Выйти*/
    {
        Session::flush();
        return redirect(route('home'));
    }

    public function mf_admin_auth(Request $request) /*Форма
авторизации Адмика*/
    {
        $id_user = 0;
        if (!empty(session('users.id'))) {
            $id_user = session('users.id');
        }

        $body = '';
        $name = 'Вход администратора';

        $url= route('auth.admin.set');
        $method ='post';

        $buttons_auth = view('t_widget.components.button')-
>with([
            'url' =>  $url,
            'name' => 'Войти',
            'type' => 'submit'
        ]);
        $buttons_auth = $buttons_auth->render();
```

```
        $buttons = $buttons_auth ;

        $auth_form = '<form class="border border-light p-5">
                            <input type="text"
id="defaultLoginFormEmail" name="login" class="form-control mb-4"
placeholder="Логин" required minlength="5">
                            <input type="password"
id="defaultLoginFormPassword" name="pass" class="form-control mb-
4" placeholder="Пароль" required minlength="4">
                        </form>';
        $body .=  $auth_form;

        $html = view('t_widget.modal_form', compact('url',
'name', 'body', 'buttons','method'));
        $html = $html->render();
        return response()->json(['mf_form' => $html, 'show_modal'
=> true]);
    }

    public function mf_admin_auth_set(Request $request)
/*Авторизация Админа*/
    {
        Session::flush();
        if (Request('login') == 'admin') {
            if (Request('pass') == 'admin'){
                Session::put('isAdmin', 'Admin');
                Session::put('users.id',-1);
            }
        }

        return redirect(route('home'));
    }

}
```

2) BookingController

```
<?php

namespace App\Http\Controllers\Booking;

use ArrayObject;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use App;
use App\LookUps;
use App\Http\Helpers\MyMail;
use Illuminate\Support\Facades\File;
use Illuminate\Support\Facades\Session;

class BookingController extends Controller
{
    public function booking_index(Request $request) /* страница
броней*/
    {
        $id_user = 0;
        if (!empty(session('users.id'))) {
```

```
            $id_user = session('users.id');
        }

        if (Session::has('users.type') and
(Session::get('users.type') == 'renter')) {
            $booking =
App\ExecProcedure::execProc('p_booking_sel', [$id_user]);
        } else
            $booking =
App\ExecProcedure::execProc('p_booking_owner_sel', [$id_user]);

        return view('page.booking.booking_index',
compact('booking'));
    }

    public function cancel(Request $request) /* страница броней*/
    {
        $id_user = 0;
        if (!empty(session('users.id'))) {
            $id_user = session('users.id');
        }
        $id_booking = Request('id_booking');

        $booking =
App\ExecProcedure::execProc('p_booking_cancel', [$id_booking,
$id_user]);

        return view('page.booking.booking_index',
compact('booking'));
    }

    public function check_date(Request $request) /* Запрос
проверки верных дат бронирования*/
    {
        $id_user = 0;
        if (!empty(session('users.id'))) {
            $id_user = session('users.id');
        }
        $id_apart = 0;
        $date_from = 0;
        $date_to = 0;
        $res =0;
        $data = Request('data_form');
        for ($i = 1; $i <= count($data) - 1; $i++) {
            $val = $data[$i];
            if ($val['name'] == 'apart_id') {
                $id_apart = $val['value'];
            }
            if ($val['name'] == 'date_from') {
                $date_from = $val['value'];
            }
            if ($val['name'] == 'date_to') {
                $date_to = $val['value'];
            }
        }

        $booking =
App\ExecProcedure::execProc('p_aparts_check_date', [$id_apart,
```

```
$date_from, $date_to]);
        foreach ($booking as $item) {
            $res = $item->result;
            $date1 = $item->date_start;
            $date2 = $item->date_end;
        }
        if ($res > 0) {
            $res = '<p class="note note-
danger"><strong>Внимание:</strong> Бронирование не возможно!<br>В
указанный период помещение занято (с '.$date1.' по '.$date2.'),
укажите другие даты</p></p>';
        } else {
            $res = '';
        }
        return response()->json(['res' => $res]);
    }

}
```

3) MessageController

```
<?php

namespace App\Http\Controllers\Message;

use ArrayObject;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use App;
use App\LookUps;
use App\Http\Helpers\MyMail;
use Illuminate\Support\Facades\File;
use Illuminate\Support\Facades\Session;

class MessageController extends Controller
{
    public function write_msg_email(Request $request) /* создания
сообщения*/
    {
        $id_user = 0;
        if (!empty(session('users.id'))) {
            $id_user = session('users.id');
        }

        $email = Request('email');

        return view('template.emails.form_write_email',
compact('email'));
    }

    public function write_msg_email_send(Request $request) /*
отправка сообщения*/
    {
        $id_user = 0;
        if (!empty(session('users.id'))) {
            $id_user = session('users.id');
        }
        $subject = Request('subject') ;
```

```
            $message = Request('message') ;
            $email   = Request('email');

            $is_send_mail = MyMail::sendMail($email, $subject,
$message);

            return redirect(url()->previous());
        }

    }
```

4) PremisesController

```
    <?php

    namespace App\Http\Controllers\Premises;

    use ArrayObject;
    use Illuminate\Http\Request;
    use App\Http\Controllers\Controller;
    use App;
    use App\LookUps;
    use App\Http\Helpers\MyMail;
    use Illuminate\Support\Facades\File;

    class PremisesController extends Controller
    {
        public function premises_add(Request $request) /*окно
    добавления помещения*/
        {
            $id_user = 0;
            $aparts_id = 0;
            $street_id = 0;
            if (!empty(session('users.id'))) {
                $id_user = session('users.id');
            }
            $city = Request('select_city');
    //        if ($city == 1){
                $city = 'Челябинск';
    //        } else{
    //            $city = '';
    //        }
            if (!empty(Request('aparts_id'))) {
                $aparts_id = Request('aparts_id');
            }

            $aparts = App\ExecProcedure::execProc('p_aparts_sel',
    [$id_user, 0,0,$aparts_id,'2020-01-01','2020-01-01',0,0,0]);
            if (!empty($aparts)){
                $aparts = collect($aparts[0])->toArray();
                $street_id = $aparts['street_id'];
            }

            $list_street =
    App\ExecProcedure::execProc('p_lkp_street', [$id_user,
    $street_id]);

            return view('page.premises.premises_add', compact('city',
```

```
'list_street','aparts'));
    }

    public function premises_mf_add(Request $request) /*
модальное окно выбора типа помещения для добавления*/
    {
        $id_user = 0;
        if (!empty(session('users.id'))) {
            $id_user = session('users.id');
        }
        $body = '';
        $name = 'Укажите город';

        $url= route('premises.add');
        $method ='get';
        //   $product_types =
App\ExecProcedure::execProc('p_product_types_SEL', ['id_user',
'All']);

        $buttons_auth = view('t_widget.components.button')-
>with([
            'url' =>  $url,
            'name' => 'Далее',
            'type' => 'submit'
        ]);
        $buttons_auth = $buttons_auth->render();

        $buttons = $buttons_auth;

        $select_type ='<select class="form-control browser-
default custom-select " name="select_city" id="select_city"
required="">
                    <option value="" disabled="">Укажите
город</option>
                    <option value="1"
selected>Челябинск</option>
                </select>';

        $body .=  $select_type;

        $html = view('t_widget.modal_form', compact('url',
'name', 'body', 'buttons','method'));
        $html = $html->render();
        return response()->json(['mf_form' => $html, 'show_modal'
=> true]);
    }

    public function premises_add_send(Request $request)
/*Добавление помещения в базу*/
    {
        $id_user = 0;
        if (!empty(session('users.id'))) {
            $id_user = session('users.id');
        }
        $city = Request('select_city');
        if ($city == 1){
            $city = 'Челябинск';
        }
```

```
        else{
            $city = '';
        }
        $street_id = Request('select_street');
        $aparts_id = Request('aparts_id');
        $number =  Request('number_house');
        $letter =  Request('letter_house');
        $n_room =  Request('rooms');
        $letter_apart=  Request('letter_apart');
        $number_apart=  Request('number_apart');
        $area=  str_replace(',','.',Request('area')) ;
        $kitchen=  str_replace(',','.',Request('kitchen'));
        $time_in=  Request('time_in');
        $time_out=  Request('time_out');
        $description = htmlspecialchars(Request('description'));
        $couchette_count  = Request('couchette_count');
        $path_img_house = '';
        $name_file = '';
        $path_to_db = Request('path_to_db');
        if($request->hasFile('image_house')) {
            $img = $request->file('image_house');
            $ext = $img->getClientOriginalExtension(); //
расширение файла
            $name_file = App\GlobalProcedure::getFileNameRandom()
. '.' . $ext; // Новое имя файла с расширением
            $path_to_db = $street_id.'-'.$number.$letter. '-
'.$name_file;
            $path_img_house = public_path() . env('IMG_HOUSE');
            $img->move( $path_img_house,$path_to_db);
        }

        $house = App\ExecProcedure::execProc('p_house_set',
[$id_user, $street_id, $number, $letter, $path_to_db]);
        foreach ($house as $row){
            $house_id = $row->house_id;
        }

        $aparts = App\ExecProcedure::execProc('p_aparts_set', [
            $id_user,
            $aparts_id,
            $house_id,
            $id_user,
            $n_room,
            $letter_apart,
            $number_apart,
            $area,
            $kitchen,
            $time_in,
            $time_out,
            $description,
            $couchette_count
            ]);

        $aparts_id = collect($aparts[0])->toArray();
        return
redirect(route('premises.show',$aparts_id['aparts_id']));
    }
```

```
    public function premises_show($id) /*Просмотр апартаментов*/
    {
        $apart_id = $id;
        $id_user  = 0;
        if (!empty(session('users.id'))) {
            $id_user = session('users.id');
        }

        $apart  =  App\ExecProcedure::execProc('p_aparts_sel',
[$id_user, 0, 0, $apart_id, '2020-01-01', '2020-01-01',0,0,0]);

        return view('page.premises.premises_show',
compact('apart', 'left_menu'));
    }


    public function mf_premises_deleted(Request $request)
/*Удаление апартаментов (мод. форма)*/
    {
        $id_user = 0;
        $aparts_id = 0;
        if (!empty(session('users.id'))) {
            $id_user = session('users.id');
        }
        $data_from = Request('data_form');
        $data_from = $data_from[1];

        $aparts_id = $data_from['value'];


        $body = 'Подтвердите удаление апартаментов';
        $name = 'Удаление апартаментов';
        $body.= '<input type="hidden" name="aparts_id"
value="'.$aparts_id.'">';
        $url= route('premises.deleted');
        $method ='post';

        $buttons_auth = view('t_widget.components.button')-
>with([
            'class'=>'w-200px',
            'url' =>  $url,
            'name' => '<i class="fas fa-ban mr-2 text-danger mr-
2"></i>Удалить',
            'type' => 'submit'
        ]);
        $buttons_auth = $buttons_auth->render();
        $buttons = $buttons_auth;

        $html = view('t_widget.modal_form', compact('url',
'name', 'body', 'buttons','method'));
        $html = $html->render();
        return response()->json(['mf_form' => $html, 'show_modal'
=> true]);
    }


    public function premises_deleted(Request $request) /*Удаление
апартаментов*/
```

```
    {
        $id_user = 0;
        $aparts_id = 0;
        if (!empty(session('users.id'))) {
            $id_user = session('users.id');
        }

        if (!empty(Request('aparts_id'))) {
            $aparts_id = Request('aparts_id');
        }

        $aparts_del =
App\ExecProcedure::execProc('p_aparts_delete', [$aparts_id]);

        return redirect(url()->previous());
    }

}
```

5) RentController

```
<?php

namespace App\Http\Controllers\Premises\Rent;

use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use App;
use App\Http\Helpers\MyMail;
use Illuminate\Queue\RedisQueue;
use Illuminate\Support\Facades\Session;

class RentController extends Controller
{
    public function rent_index(Request $request) /* открыть
главное окно аренды*/
    {
        $apart_start = 0;
        $apart_count = 100;
        $id_user = 0;
        $street_id = 0;
        $couchette_count = 0;
        $price_to = 100000000;
        $date_from = date("Y-m-d");
        $date_to = date('Y-m-d', mktime(0, 0, 0, date('m'),
date('d') + 7, date('Y'))); // дата через 7 дней

        if (!empty(session('users.id'))) {
            $id_user = session('users.id');
        }
        if (!empty(Request('date_from'))){
            $date_from = Request('date_from');
            $date_to = Request('date_to');
        }
        if (!empty(Request('select_street_with_apart'))){
            $street_id = Request('select_street_with_apart');
        }
        if (!empty(Request('price_to'))){
```

```
            $price_to = Request('price_to');
        }
        if (!empty(Request('couchette_count'))){
            $couchette_count = Request('couchette_count');
        }

        Session::put('date_rent_start', $date_from);
        Session::put('date_rent_end', $date_to);

        $apart_list  =
App\ExecProcedure::execProc('p_aparts_sel', [$id_user,
$apart_start, $apart_count, 0, $date_from, $date_to, $price_to,
$street_id, $couchette_count]);
        $list_street_with_apart =
App\ExecProcedure::execProc('p_lkp_street_with_apart', [$id_user,
$street_id]);
        return view('page.premises.rent.rent_index',
compact('apart_list','list_street_with_apart','date_from','date_t
o','price_to','couchette_count'));
    }

    public function rent_form_open($id_apart) /* открыть форму
бронирования*/
    {
        $id_user = 0;       $date_from = date("Y-m-d");
        $date_to = date('Y-m-d', mktime(0, 0, 0, date('m'),
date('d') + 7, date('Y'))); // дата через 7 дней

        $day_count = (strtotime (Session::get('date_rent_end'))-
strtotime (Session::get('date_rent_start')))/(60*60*24);

        if (!empty(session('users.id'))) {
            $id_user = session('users.id');
        }
        if(Session::has('users.type') and
(Session::get('users.type') == 'renter')){
            $apart  =
App\ExecProcedure::execProc('p_aparts_sel', [$id_user, 0, 0,
$id_apart, $date_from, $date_to, 0, 0, 0]);
            return view('page.premises.rent.rent_form',
compact('apart', 'day_count'));
        }
        else{
            $msg = 'Для бронирования, авторизуйтесь под
"Арендодателем"';
            $msg_error = setcookie( 'msg_error',  $msg );
            // $cookie = Cookie::forget('vendor');
            return  redirect($_SERVER['HTTP_REFERER']);
        }
    }

    public function booking_set(Request $request) /* Бронирование
апартаментов*/
    {
        $id_user = 0;
        if (!empty(session('users.id'))) {
            $id_user = session('users.id');
        }
```

```
        // ToDo отправка сообщения для бронирования такси. СКрыть
кнопку Бронь! Если квартира забронирована

        $isTaxi     = Request('isTaxi');
        $apart_id   = Request('apart_id');
        $date_from  = Request('date_from');
        $date_to    = Request('date_to');
        $time_in    = Request('time_in');
        $time_out   = Request('time_out');

        $booking  =  App\ExecProcedure::execProc('p_booking_set',
[$apart_id, $date_from, $date_to, $id_user]);

        $msg = 'Апартаменты забронированы';
        $msg_error = setcookie( 'msg_error',  $msg );

        return view('page.booking.booking_index');
    }
}
```

6) ProfileController

```
<?php

namespace App\Http\Controllers\Profile;

use Illuminate\Http\Request;
use App\Http\Controllers\Controller;
use App;

use App\Http\Helpers\VisualElements;

class ProfileController extends Controller
{
    public function index($type_user_open, $id_user_open) /*
открыть профиль сотрудника */
    {
        $id_user = 0;
        if (!empty(session('users.id'))) {
            $id_user= session('users.id');
        }
        $user_type = '';
        if (!empty(session('users.type'))) {
            $user_type = session('users.type');
        }

        if ($type_user_open == 'owner'){
            $apart_list = App\ExecProcedure::execProc
('p_aparts_owner',[$id_user_open]);
        }
        else{
            $apart_list = App\ExecProcedure::execProc
('p_aparts_owner',[0]);
        }

        $profile_list = App\ExecProcedure::execProc
('p_profile_sel',[$id_user_open, $type_user_open,$id_user,
$user_type]);
```

```
        return view('page.profile.profile_index',
compact('apart_list', 'profile_list', 'type_user_open'));
    }

    public function user_update(Request $request) /* Обновление
информации о пользователе */
    {   $user_id = 0;
        $type_user = '';
        $FIO = '';
        $email = '';
        $phone = 0;
        $data = Request('data_form');
        for ($i = 1; $i <= count($data) - 1; $i++) {
            $val = $data[$i];
            if ($val['name'] == 'user_id') {
                $user_id = $val['value'];
            }
            if ($val['name'] == 'type_user') {
                $type_user= $val['value'];
            }
            if ($val['name'] == 'FIO') {
                $FIO = $val['value'];
            }
            if ($val['name'] == 'email') {
                $email = $val['value'];
            }
            if ($val['name'] == 'phone') {
                $phone = $val['value'];
            }
        }

        $user_upd = App\ExecProcedure::execProc
('p_auth_upd',[$user_id, $type_user, $email, $phone, $FIO]);

        return response()->json(['res' => $user_upd]);
    }
}
```

7) UsersController

```
<?php


namespace App\Http\Controllers\Users;


use Illuminate\Http\Request;

use App\Http\Controllers\Controller;

use App;


use App\Http\Helpers\VisualElements;


class UsersController extends Controller

{
```

```
    public function index(Request $request) /* открыть главную
страницу */
    {
        $id_user = 0;
        if (!empty(session('users.id'))) {
            $id_user = session('users.id');
        }

        $user_list = App\ExecProcedure::execProc
('p_users_get',[$id_user]);
        foreach ($user_list as $row){
            $u_list[] = [ 'id'=>$row->id,
                          'email'=>$row->email,
                           'fio'=> $row->fio,
                'img'=>$row->img,
                'stars'=>VisualElements::getStars($row-> stars)];
        }
        return view('users.users_index', compact('u_list'));
    }
    public function mf_settings(Request $request) /* Настройка
прав пользователя(админка) */
    {
        $id_user = 0;
        if (!empty(session('users.id'))) {
            $id_user = session('users.id');
        }
        $body = '';
        $name = 'Настройка пользователя';

        $url= route('auth.mf_set');
        $method ='post';

        $buttons_auth = view('t_widget.components.button')-
>with([
            'url' =>  $url,
            'name' => 'Войти',
            'type' => 'submit'
        ]);
```

```
        $buttons_auth = $buttons_auth->render();

        $buttons = $buttons_auth ;

        $auth_form =  view('template.base_form.auth')->with([
            'name' => 'vvv',
            'type' => 'submit'
        ]);
        $body .=  $auth_form;

        $html = view('t_widget.modal_form', compact('url',
'name', 'body', 'buttons','method'));
        $html = $html->render();
        return response()->json(['mf_form' => $html, 'show_modal'
=> true]);
    }
    public function mf_user_roles(Request $request) /* Настройка
ролей пользователя (админка) */
    {
        $user = Request('user');

        $body = '';
        $name = 'Настройка ролей пользователя';

        $url= route('roles.edit_set');
        $method ='post';
        $role_list = App\ExecProcedure::execProc('p_sprav_roles',
[$user]);

        $buttons_auth = view('t_widget.components.button')-
>with([
            'url' =>  $url,
            'name' => 'Сохранить',
            'type' => 'submit'
        ]);
        $buttons = $buttons_auth ;

        $auth_form =  view('users.users_roles')->with([
            'role_list' => $role_list,
```

```
            'type' => 'submit'
        ]);
        $body .='<input name="id_user" type="hidden"
value="'.$user.'">';
        $body .=  $auth_form;

        $html = view('t_widget.modal_form', compact('url',
'name', 'body', 'buttons','method'));
        $html = $html->render();
        return response()->json(['mf_form' => $html, 'show_modal'
=> true]);
    }
    public function mf_user_roles_set(Request $request) /*
Сохранение ролей пользователя (админка) */
    {
        $r = Request('data_form');
        $role_id =$r[2];
        $right_id = $r[0];
        if ($r[1] == 'true' ){
            $new_right = App\ExecProcedure::execProc
('p_roles_rights_set',[$role_id, $right_id, 1]);
        }
        else{
            $new_right = App\ExecProcedure::execProc
('p_roles_rights_set',[$role_id, $right_id, 0]);
        }
        return response()->json(['show_modal' => false]);
    }
}
```